Bruce Christianson
Bruno Crispo
James A. Malcolm
Michael Roe (Eds.)

LNCS 2845

Security

10th International Worksh
Cambridge, UK, April 2002
Revised Papers

Lecture Notes in Computer Science 2845

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Bruce Christianson Bruno Crispo
James A. Malcolm Michael Roe (Eds.)

Security Protocols

10th International Workshop
Cambridge, UK, April 17-19, 2002
Revised Papers

Springer

Volume Editors

Bruce Christianson
James A. Malcolm
University of Hertfordshire, Computer Science Department
Hatfield AL10 9AB, UK
E-mail: {b.christianson,J.A.Malcolm}@herts.ac.uk

Bruno Crispo
Vrije Universiteit
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
E-mail: crispo@cs.vu.nl

Michael Roe
Microsoft Research Ltd.
7 J.J. Thomson Avenue, Cambridge CB3 0FB, UK
E-mail: mroe@mircosoft.com

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): E.3, F.2.1-2, C.2, K.6.5, J.1, K.4.1, D.4.6

ISSN 0302-9743
ISBN 3-540-20830-5 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 10976111 06/3142 5 4 3 2 1 0

Preface

Once again we bring you the proceedings of the International Workshop on Security Protocols. It seems hard to believe that we have reached the tenth event in this annual series.

This year our theme was "Discerning the Protocol Participants." Security protocols are usually described in terms of the active participants – Alice computes *foo* and sends it to Bob. However most security protocols also include off-line participants, which are not synchronously involved in the exchange of messages: a bank may participate on behalf of a customer, and an arbiter may subsequently be asked to interpret the meaning of a run.

These silent partners to the protocol have their own security policies, and assumptions about identity, authorization and capability need to be re-examined when the agenda of a hidden participant may change.

We hope that the position papers published here, which have been rewritten and rethought in the light of the discussions at the workshop, will be of interest, not just for the specific contributions they make but also for the deeper issues which they expose. In order to identify these issues more clearly, we include transcripts for some of the discussions which took place in Cambridge during the workshop. What would you have liked to add? Do let us know.

As in past years, these proceedings also include a transcript of the keynote address given by Roger Needham. Alas, this is the last time. Roger's death during the preparation of these proceedings represents a loss, not only to us in the security community but indeed to the whole of computer science, of a magnitude that we are only just beginning to discern. In these proceedings, as in life, he has the last word.

Our thanks to Sidney Sussex College, Cambridge for the use of their facilities, to Lori Klimaszewska of the University of Cambridge Computing Service for transcribing the audio tapes (in which fine grain cement at Wapping nearly proved a sticking point for concurrency) and to Johanna Hunt at the University of Hertfordshire for her assistance in editing the resulting Heraclitian texts.

July 2003

Bruce Christianson
Bruno Crispo
James Malcolm
Michael Roe

Previous Proceedings in This Series

The proceedings of previous International Workshops on Security Protocols have also been published by Springer-Verlag as Lecture Notes in Computer Science, and are occasionally referred to in the text:

9th Workshop (2001), LNCS 2467, ISBN 3-540-44263-4
8th Workshop (2000), LNCS 2133, ISBN 3-540-42566-7
7th Workshop (1999), LNCS 1796, ISBN 3-540-67381-4
6th Workshop (1998), LNCS 1550, ISBN 3-540-65663-4
5th Workshop (1997), LNCS 1361, ISBN 3-540-64040-1
4th Workshop (1996), LNCS 1189, ISBN 3-540-63494-5

Table of Contents

Introduction (Transcript)

Bruce Christianson

University of Hertfordshire

Each year we have a theme, and each year people give talks about either the previous year's theme or next year's theme. This year the theme is discerning the protocol participants. There are usually several people who have an interest in a particular run of a protocol besides those who are actually sending and receiving the messages. When we come to do the proceedings we'll try and pretend that everything that was said has something to do with this. Today we're looking mainly at authentication. There are some people who are going to try to convince us that authentication is harmful, there are some who are going to talk about why authentication is a good thing that doesn't actually involve knowing who people are in real life, there are some people who are going to argue that authentication doesn't involve trusted third parties and that trusted third parties are actually bad for you, there are other people who will talk about why trusted third parties are jolly good things to have provided you don't use them for any of the things that they're usually sold to you for.

For the benefit, partly for those of us who have to do the transcripts, but also for those speakers who actually have something to say, could we please follow the ten minute rule. When you first stand up you have ten minutes during which you can only be interrupted if the audience either genuinely can't hear you, or can't understand what you're trying to say. Please don't interrupt on the basis of disagreement or theological principle for the first ten minutes. Conversely if you are introducing your topic, you've got ten minutes to get across the big idea, and after that you should expect to make progress, not necessarily slowly, but certainly not in the direction you had in mind. The purpose of the session is mainly to try and discuss the issues that are interesting. We do actually all read the papers quietly when we've got a moment to ourselves. You will have the chance to revise your position paper before it's published in the proceedings, and we hope in future to give you an unedited transcript of the discussions to help you with that process. We then, very carefully, revise the transcripts with a cautious eye on the libel laws in a manner similar to that allegedly used by (deleted). My ten minutes are up, so without further ado I therefore ask Roger to, as has become traditional, give us a keynote address to put us in the right frame of mind.

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, p. 1, 2004.

Keynote Address

Roger Needham

Microsoft Research

Well good morning. When I saw the ostensible title of this workshop I asked Bruce what it meant, because it didn't convey anything to me whatsoever. I gather that what it really means is, when some protocol occurs, finding out all of the agencies that have any influence whatever over what happens. Now this is a tall order because if influencing what happens includes being an attacker, it includes, in a general way, everyone. So maybe one should say that the art of designing these protocols is to ensure that attackers don't influence what happens. You can't achieve that either, because influencing what happens includes causing retries by deleting messages. I would suggest that in some interpretation of what Bruce has said, the participants in the protocol are everybody, end of story, close workshop, but this is perhaps not quite what he meant. He means things like trusted third parties or semi-trusted third parties, but as soon as you get on to the people who provide the communication then this seems to be getting a bit dubious. The whole point of being a communications provider is that you shouldn't be a participant, you should be indiscernible.

It also occurred to me that there is a complimentary subject which is obscuring the participants in the protocol. You could say that the goal of protocol design is to make it appear that there aren't any more participants than the prime participants — generically, Alice and Bob. The purpose should be to make it appear that there isn't anybody else, even if there is. But I think one can quickly get into a theological state about this sort of discussion, and I don't want to do that.

The thing that does seem to me to be a trend at the moment is to accept that there isn't going to be any infrastructure that is in a general way everywhere. At one time it had tended to be assumed that there would be a public key infrastructure, to which everybody who was going to participate in anything, anywhere, belonged. If you said that such a thing was so impossible that basing your work on the assumption that it existed was not sensible, one tended to be described as some sort of luddite or unbeliever, and generally regarded with scorn.

What seems to have changed is that it's now much more widely realised that there is not going to be, and there never will be, a universal infrastructure that everyone subscribes to. The thing that really makes it become obvious, is the existence of a huge multiplicity of typically mobile devices. When you go back to an earlier workshop and think of Frank Stajano[1], with his resurrecting

[1] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks", LNCS 1796, pp 172-182.

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 2–4, 2004.

duckling protocol, where the participants are simply objects got out of boxes, and identified by where you get them out of the box rather than anything else, you clearly are not going to have a universal infrastructure, and it is completely erroneous to think you ever will. And this has given rise to interesting work trying to explore the scope and limits of what you can do in the absence of the thing you can't have. And I think this is greatly to be encouraged. It's quite wrong to say we don't have a universal infrastructure therefore we can't do anything, although it's probably the case that there are some things you could only do if you did have a universal infrastructure.

The interesting research question is, what are the boundaries between the two? What can you achieve with the kind of background it's reasonable to expect? This is a type of research which is curiously not often done. I can give an example from a non security field. If you're doing distributed computing implementations, everyone knows it's nice to do them with idempotent protocols if you can, that is to say a protocol where if you don't get a reply to your message you just send it again, and the whole thing is guaranteed to work as well. Such things are very easy to write. I have never seen any characterisation of the class of problem that you can expect to solve with idempotent protocols. If, given a system, you could readily calculate and say "oh yes, this is one where you can live with idempotent protocols", or "no, this is one where you can't, it's therefore not sensible to try", it would be good.

Another example of a somewhat similar character is where you have replicated data, or data that is meant to be replicated, but you can't, or didn't, take steps to do updates transactionally so that, in practice, the things that are alleged to be written never really are. Twenty odd years ago I tried to characterise a set of applications that you could do that way, and miserably failed. I don't think anybody has succeeded since. So I applaud any endeavours to set limits to what you can do in the absence of the infrastructure you can't have.

Now, I would applaud research directed at really obscuring who the participants are. This after all is what electronic cash protocols are supposed to be for, making sure the participants are not known, that's what cash is about.

Kai Rannenberg very recently made a nice observation. He did a piece of work in the Microsoft lab here where he put an HTTP server in the SIM of his mobile phone, and you can make this participate in transactions. He had a cool demo on the basis that you've ordered something over the internet, and on your phone appears the remark, do you really want to pay £27.50 for so and so, and he would say yes or no. He points out that if it's a prepayment phone, the recipient of the money doesn't know where he got it from because that's what prepayment phones are about — so maybe the way you make electronic cash is with prepaid cell phones with extra software in the SIM.

So I will leave it at that. Two messages: characterise what you can do in the world as it really is, and work on failing to disclose the participants in transactions as much as you work on succeeding in disclosing them. Thank you.

Who has a question?

Matt Blaze: Sorry Roger, I saw a dead horse and I just can't resist the opportunity to flog it. You mentioned public key infrastructure as your example of the infrastructure that's harder to do than most of us thought, and therefore you can't have it. I'm going to offer a slightly different view of that. Is it possible that in fact public key infrastructure would be quite easy to do, but we don't have it simply because no-one wants it? Certainly the key distribution aspects of public key infrastructure seem very straightforward, they don't seem to be any more difficult than anyone originally anticipated, systems like DNS appear to do it. But the idea of a global trust authority, Verisign's marketing notwithstanding, appears to be relatively useless. So is it possible, not that applications that need global key infrastructure are failing, but rather that those applications simply don't exist?

Roger Needham: I think that's very possible. I certainly agree with distinguishing the mechanics of a public key infrastructure from the global trust authority. They are quite different. In practice, inside organisations such as large corporations, you can have a public key infrastructure, and you can have a trust authority, but it only works inside Microsoft, or IBM, or Unilever, or what have you.

Weak Authentication: How to Authenticate Unknown Principals without Trusted Parties

Jari Arkko and Pekka Nikander

Ericsson Research NomadicLab,
02420 Jorvas, Finland
{Jari.Arkko,Pekka.Nikander}@nomadiclab.com

Abstract. This paper discusses "weak authentication" techniques to provide cryptographically strong authentication between previously unknown parties without relying on trusted third parties.

1 Introduction

Introducing security mechanisms into environments with little or no existing security, such as the Internet, has proven to be a challenging task. This is mainly due to the practical problems, such as demands for security infrastructure and configuration tasks associated with the used mechanisms. These problems are significant barriers to security deployment, particularly for large networks.

The purpose of this paper is to study "weak authentication" techniques that attempt to bypass some of these problems. In this paper, we use the term "weak authentication" to denote cryptographically strong authentication between previously unknown parties without relying on trusted third parties. We consider this kind of authentication "weak", since the parties involved do not have any "real" information about the identity of their peer.

Thus, at first sight it seems impossible to **authenticate** the identity of the parties in the traditional sense. Surprisingly, however, a number of different techniques can provide some security. None of the techniques are as secure as traditional "strong" identity authentication. It is impossible or at least extremely unlikely for attackers to be able to breach strong authentication techniques. In weak authentication, the probability of a successful attack is also quite small, but not zero.

As discussed earlier by Anderson [2], it is crucial to understand the economic drivers and impacts of various security solutions. Our main research hypothesis is that protocol design should take place with a full understanding of the economic impacts for attacks and defenses, and the residual vulnerability probabilities involved. A secondary hypothesis is that under such analysis, weak authentication techniques offer a better security model for some applications than traditional security methods.

This paper is organized in four parts. First, we need to give an overview of potential applications for weak authentication, and discuss where weak authentication may or may not be applicable. This is done in Sect. 2. Section 3 discusses a

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 5–19, 2004.

categorization and classification of weak authentication techniques. Section 4 reviews specific techniques for weak authentication and discusses their properties. For instance, a successful attack against a weak authentication method typically requires specific conditions relating to the network topology and the locations of the victims and the attacker over time. Finally, Sect. 5 discusses some tools for understanding the economic aspects and uncertaintities relating to the the residual vulnerabilities in weak authentication methods.

2 Potential Applications

There are many existing and planned applications of weak authentication in some form. Personal area networks are a typical application, such as when we need to connect a mobile phone to a Bluetooth headset. The Secure Shell (SSH) protocol applies weak authentication principles for authenticating servers to users [22]. Users can authenticate their multimedia call signaling using weak authentication in the Session Initiation Protocol (SIP) protocol [18]. The use of weak authentication in providing security for multi-homing, mobility, and other network layer tasks is also being studied [12].

However, weak authentication is only suitable for some applications. In the following, we discuss first some fundamental issues that relate to this suitability.

Some applications require that a real-world identity is presented. Weak authentication is typically more suited for dealing with ephemeral identities. Likewise, weak authentication is less suitable for applications that have real-world impacts. However, a significant number of applications do not have a need for real-world identities, and do not directly affect the real world. Multi-homing, routing, and mobility are typical examples of functions that do not have real-world effects.

The level of protection achieved with weak authentication is also a determining factor. In some special cases – such as for the imprinted duckling described by Stajano and Anderson [19] – a security level comparable to strong authentication is reached. In other cases, it is necessary to compare the remaining vulnerabilities left after weak authentication to those present in the system for other reasons.

In some applications, the cost of strong authentication may exceed the benefits. For instance, Mobile IPv6 route optimization does not justify the introduction of a global Public Key Infrastructure for the sole purpose of authenticating nodes participating in these optimizations. Similarly, some light-weight and low-cost devices may become bulky and impractical if user input devices or tamper-resistance is required as a part of the security solution.

In some other applications that use weak authentication, the cost of attacks to an attacker may exceed the value of the transactions. These applications can, in many cases, be viewed as secure enough.

3 Categories of Techniques

Weak authentication can't be based on pre-shared secrets, as the peers by definition do not have a priori security information about each other. Also, enrollment to an infrastructure is not possible, since that would constitute a trusted third party (or parties), and would require configuration. Traditional Public Key Infrastructure (PKI) or the proposed IETF Authentication, Authorization and Accounting (AAA) infrastructure is therefore out of the question.

One may then ask what is left, and how can any authentication be performed at all? It turns out that there are still several properties that we can use for weaker forms of "authentication" such as *spatial separation*, *temporal separation*, *asymmetric war* on costs, *application semantics*, and *combined* or *transitive* techniques. We discuss each of these separately below.

3.1 Spatial Separation

Spatial separation refers to the ability of a node to ensure that its peer is on a specific communications path. In its simplest form, spatial separation reduces to the requirement of a physical contact, such as the imprinting process proposed for ad hoc wireless networks by Stajano and Anderson [19,1]. Assuming the underlying communications network is trustworthy to an extent, we can also use challenge-response protocols to ensure that the peer is on a specific path. In a more general setting, a node may view different links and paths as having different degrees of trustworthiness. More than one paths can also be used to further enhance the situation.

3.2 Temporal Separation

By temporal separation we refer to the general ability of the peers to relate communications at time t_1 to some earlier communications at time t_0. That is, there exists some guarantee that the node we talk to at time t_1 is the same one as we talked to at t_0. The guarantees are typically not complete, i.e., they exist only with respect to some assumptions, such as that there have been no man-in-the-middle attackers present at time t_0.

3.3 Asymmetric Costs

As noted by Varian [20] and Anderson [2], economic impacts and interests are often major factors in designing security mechanisms[1]. A troubling observation is that usually it is easier for the attacker to find a single security hole than for the defender to block all holes.

Fortunately, this asymmetric situation can sometimes be reversed and used to the defenders' advantage. This is possible when an individual defender can

[1] Unfortunately, these factors are often calculated from the point of view of a vendor and not the user.

spend a small amount of resources, which then are multiplied on the attacker's side.

The defender can make it harder for the attacker to find the right target. We can also force the attacker to use more resources on an attack even he knows the right target. As an example of the latter, even unauthenticated challenge-response mechanisms work well in some cases [4,18]. An attacker would have to reveal his location (address) in the network in order to be able to receive the challenge. If the attacker has only a limited number of network locations available for him, he may not be willing to sacrifice them for the sake of being able to attack. More seriously, revealing a network location may also lead to real-world implications for the attacker, such as the network operator or the police taking action against him.

3.4 Application Semantics

Certain applications may offer particular semantics that can be employed to produce weak forms of authentication. For instance, some proposed methods for authorizing Mobile IPv6 route optimization rely on embedding cryptographic information in IP addresses [15,17].

3.5 Combined and Transitive Techniques

The techniques presented above can be combined. For instance, spatial separation techniques can be combined with temporal separation. An initial physical contact can be used to exchange keys, and leap-of-faith techniques can be used to rely on these later, even over insecure channels.

Web-of-trust models and information from neighbours could potentially be used together with weak authentication techniques to reduce the likelihood of successful attacks. For instance, assume two nodes a and b each have contacted a server c earlier at times t_a and t_b, respectively. Further assume that a and b both can ensure that the node c contacted earlier is still the same c as they are talking to now. This is possible, for instance, if the server gave its public key at the first contact. Then, if a and b trust each other, they verify that they have the same key. Both nodes can then be assured that the server's key has been the same since $min(t_a, t_b)$.

4 Some Concrete Techniques

4.1 Anonymous Encryption

Anonymous encryption is based on temporal separation. In this method, unauthenticated Diffie-Hellman key exchange or some equivalent protocol is run in the beginning of all communication sessions. Here the time period $[t_0...t_1]$ extends only to the length of a single session. This protects the privacy and integrity of the session, assuming there were no active attackers on the communications path at the time the Diffie-Hellman procedure was executed.

Anonymous encryption reduces the likelihood of successful attacks by changing the type of attack required, by requiring attackers to capture more packets, by requiring the attackers to work more in order see communications, and by forcing attacks to be performed before the attacker knows whether a given piece of communication will be useful for it. Nevertheless, the benefits of anonymous encryption for individual users are quite small, except in a statistical sense as we will see later.

4.2 Challenge-Response

Challenge-response techniques can be used to ensure freshness and, under certain assumptions, that the peer is on a specific path towards the given address. It is therefore typically used to ensure spatial separation.

For instance, the Return Routability (RR) method [13,14,17] ensures that the peer really is in the place in the network it claims to be in, or at least on a path to it. Such properties may be useful in applications such as Mobile IPv6, which attempts to ensure that modifications made to local routing information do not break existing security properties. While RR provides a low level of protection, it should be noted that so does the current unauthenticated IP routing. Hence modifications to the local routing information can be performed if it can be proven that the node would have been able to get to the traffic anyway. Another example of challenge-response techniques is the use of cookies in TCP SYN packets [4].

This type of weak authentication reduces the risk of successful attacks in a topological sense: out of all nodes in the whole network, only a very small and specific subset can perform an attack.

4.3 Leap-of-Faith

Leap-of-faith is another method based on temporal separation, but it also encompasses aspects from spatial separation and asymmetric cost wars.

Leap-of-faith methods have been successfully employed, e.g., in the SSH protocol [22]. Here the time period [t0...t1] extends from the first contact between the peers to infinity. While in principle similar to the anonymous encryption approach, grouping all sessions under the same protection allows improved security. For instance, the first connection can typically be executed in a safe place. As an example, a Bluetooth earplug and phone can be “paired” in the privacy of the user’s home, and the pairing is subsequently secure even in environments that contain potentially malicious parties, such as in a crowded city street [19].

Furthermore, to remain undetected, an attacker must be present at the communications path each time the keys verified through leap-of-faith are being used. For example, if there has been an attacker present when the initial SSH keys have been exchanged, the attacker must be there each time the keys are being used as well. Otherwise the used software would warn the user about a changed key.

The SIP protocol [18] intends to use leap-of-faith methods for multimedia signaling, with the intention that the first call between users establishes self-signed certificates. These certificates would be related to the used identifiers, and would be stored permanently in the phones (guided by user's acceptance).

Role of User Confirmation. Typically, some kind of user confirmation is necessary for leap-of-faith to work in a secure manner. SSH, for instance, authenticates servers by leap-of-faith but the user is prompted to accept the first connection, and warned if the the keys have been changed. (Traditional identification, based on passwords, is used to authenticate the client, which means that the server can proceed unattended.) An imprinting process through a physical contact can also be thought of as a user confirmation.

User confirmation obviously requires at least a rudimentary user interface. The software making use of leap-of-faith also needs the ability to access this user interface, which may not be practical for kernel-level protocols.

Bidirectional leap-of-faith suffers from the problems of getting two user confirmations at the same time. In the case of multimedia phone calls, this problem does not exist as both users must be present for the call to be accepted in any case. In the case of leap-of-faith for general communications this is problematic, since relaxation of the user confirmation rule may create denial-of-service vulnerabilities.

Gehrmann et al have designed a secure user confirmation method to be used in personal area networks [6]. It is required that the devices have a moderately sophisticated user interface. A personal CA device can generate a hash value based on input received over a wireless interface, display this value, and allow the user to enter the same value in the other device for confirmation. We call this type of user confirmation cryptographically strong, as it can not be defeated even by an active attacker. The CA device can also take the role of reducing the need to configure combinations of devices, as the CA produces certificates that can be accepted by other devices in the same network.

State Requirements. In contrast to anonymous encryption, leap-of-faith needs to store state for a long time. Typically, the state consists of an identifier and a key. Where public keys are used, it is possible to store only a hash of the public key and allow the peer to send the public key whose hash we can now verify. Where self-signed certificates are used, even the identifier can be omitted since the hash can cover the whole certificate with the identifier inside it.

When no user confirmation is required, connection attempts (legitimate or fraudulent) from a large number of peers may lead to memory exhaustion. Furthermore, if a real-world identity, such as an IP address, is used, attackers could add just a single fraudulent entry and prevent that particular legitimate peer from establishing a connection later.

A policy is needed for accepting entries for new peers and for purging entries for some peers. Typically, such policy might favor preserving old and frequently

used peer information over recent (and possibly fraudulent) information. Alternatively, if the memory consists of just hashes, lack of memory can be compensated by reducing the length of the hashes by one bit, increasing the risk of collisions but making room for more entries.

Variants of Leap-of-Faith. A technique similar to leap-of-faith can be used with the Host Identity Protocol (HIP) [10,9,11]. The verification of the identity happens through the peer offering its identifier, public key, and a signature produced with the private key. What this method provides is a cryptographic proof that the peer really has the given identifier ID. It is therefore not possible for anyone else to claim that their identifier is ID as well. It is, however, possible for anyone to claim that their identifier is ID'. This type of weak authentication is not in general useful to signify what the node in question can be trusted to do. (However, we may already know ID and its authority through some other means.)

4.4 Properties of Identifiers

Recently developed methods can create a strong cryptographic binding between an identifier and its rightful "owner". The cryptographically generated address technique involves the generation of a public-private key pair and a hash of the generated public key [15]. The hash value is used as an identifier. The main application of this is in the area of proving the rightful owners of IPv6 addresses in Mobile IPv6 [17]. The lower host identifier part the IPv6 address is formed from the hash value while the upper part is used for routing and has topological meaning. This method could be applied in other contexts as well, such as in the protection of IPv6 Neighbour Discovery signaling [3].

This method is somewhat stronger than leap-of-faith and HIP-like mechanisms, as the identifier has a dual role for both routing to the right host and as a hash. A verified cryptographically generated address property implies that the peer really is the node who came up with the given IP address first. Since the routing prefix part of the address is not formed through a hash, this applies only to the host identifier part, i.e., the address is right but it hasn't been proven that the routing prefix is really the right one. For this reason, [17] proposes to combine the RR method with cryptographically generated addresses. RR is relatively good for ensuring that the claimed routing prefixes are correct, while cryptographically generated addresses are good for proving the host identifier claim. Together, they can cover the whole claimed IP address.

Weak authentication based on properties of identifiers reduces the likelihood of successful attacks only if the identifiers have a specific meaning for the particular application that uses this authentication. One example of such a specific meaning is the dual role of IP addresses in routing. Where no such specific meaning exists, authentication based on identifier properties reduces to the leap-of-faith method by allowing peers to verify that they are still communicating with the same node as they originally did.

4.5 Opportunistic IPsec

The definition of weak authentication precludes trusted parties. However, a trusted party may have different flavors. Traditionally, when we speak of a trusted party we mean an entity that we have an explicit security association with. Some weak authentication methods use third parties that are "semi-trusted" and do not have such a security association.

A particular approach to weak authentication has been explored in the context of recent "opportunistic security" extensions to the IPsec protocol set [16]. This approach is based on using a public infrastructure – DNS in this case – to supply keys for those parties that wish to employ weak authentication. Communication to other nodes is in the clear. Until DNS becomes secure, this approach is secure as long as one assumes that the path to DNS is free from attackers even if the path to the peer may not be. In practice, this translates roughly to the same security level as anonymous encryption.

As has been pointed out elsewhere, this approach may also lead to unexpected side-effects. In this case, there is an additional mechanism that allows DNS to specify an IPsec gateway to accommodate corporate gateways and other similar devices. An off-path attacker may be able to subvert the DNS mechanisms, e.g., by sending unsolicited responses to the victims, and lure it to use the attacker as the gateway.

5 Modelling Weak Authentication Impacts

5.1 Analysis of Anonymous Encryption

Anonymous encryption defeats passive attackers. However, if a man-in-the-middle is present on the used path, it is likely that this attacker will be able to catch and modify all packets relating to the same session. No memory is retained from one session to another. Therefore, the uncertainty related to the use of anonymous encryption does not change as time goes by. The uncertainty depends only on the probability of a man-in-the-middle attacker being on the used path. This uncertainty is only a bit smaller than the uncertainty of any kind of attackers being present.

However, this analysis considers only the point of view of an individual user. What happens if we consider the whole system view? What if everyone used anonymous encryption? We can analyze this situation by making first a few assumptions:

- Cost of scanning for an interesting transaction consists of equipment needed to monitor traffic on the path, and is 0.1 EUR per transaction.
- Cost of eavesdropping the interesting transaction consists of mainly storage space and analysis costs, and is 1.0 EUR per transaction.
- Cost of performing a man-in-the-middle attack on anonymous encryption consists of equipment needed to intercept traffic and run Diffie-Hellman in both directions. It is 10 EUR per transaction.

– There is one interesting transaction per one million transactions, or one interesting person in a population of a million.

We can now analyze the economic impacts for the three following cases:

1. There isn't anyone who uses anonymous encryption. In this case the attacker's costs are 0.1 EUR $*1,000,000 + 1$ EUR $= 100,001$ EUR.
2. Only the interesting person uses anonymous encryption. In this case the attacker's costs are 0.1 EUR $*1,000,000 + 10$ EUR $+1$ EUR $= 100,011$ EUR.
3. Everyone uses anonymous. In this case the attacker is forced to use an expensive scanning method (MitM) against everyone, raising the costs considerably: 10 EUR $*1,000,000 + 1$ EUR $= 10,000,001$ EUR.

The attacker considers these numbers as follows. The attacker must have sufficient resources to commit to the attack, and the potential benefit from the attack must be great enough to satisfy the expenditure.

For instance, if implemented widely for all HTTP connections, anonymous encryption would prevent widespread snooping, unless all communications would be routed through a device capable of a Diffie-Hellman man-in-the-middle attack for all sessions. Since the cost of such a device would be prohibitive, and the routing and delay aspects would probably give away this activity, the result is that all communications would be statistically more secure than today.

The cost of attacks and defenses must also be considered by the legitimate users. We need to consider the motives for the users to to implement defenses. Our defense is not directly useful for this particular user. Some users might not want to use the defense, particularly if it is costly. Such a situation is called the "tragedy of the commons" [8]. Fortunately, in many cases this situation can be avoided. For instance, a protocol may be designed in a standards origanization, and it is often possible to mandate a specific defense mechanism.

In conclusion, while techniques like anonymous encryption are not useful for a single individual, they may be useful for the community as a whole. This conclusion is, however, dependent on the assumption that the attacker is interested on a particular person. If the attacker is satisfied just with finding anyone to attack, the conclusion no longer applies. We should also note that the attackers and legitimate users often do not have equal resources. For instance, hand-held low-power devices have limited CPU resources, but an attacker that controls a number of virus-infected personal computers may have a large amount of CPU resources.

5.2 Analysis of Challenge-Response

Challenge-response mechanisms use spatial separation and ability of only a limited number of attackers to see the challenge. Assuming each path has some small probability (say 0.1) of having an attacker, challenge-response mechanisms leave this level of uncertainty with its authentication result. This uncertainty

can be compared to the probability of *some* path in the Internet having an attacker, which is 1. A node that does not use challenge-response and accepts commands from other nodes directly has the uncertainty of 1, while a node that uses challenge-response has an uncertainty of 0.1.

Economic analysis of challenge-response mechanisms is also possible. Consider the application of Mobile IPv6, and the use of the RR method. Assume that the attacker is interested in causing as large distributed denial-of-service attack as possible. Where would this attack be easiest to perform? Obviously at the core of the network, as the attacker could then be in a place where the number of nodes using path with support for RR is the largest. We can say that the value of the location is $nodes * capacity$, where *nodes* is the number of nodes that view this location as the path towards some useful attack address, and *capacity* is the bandwidth available at the location for the attacker.

5.3 Leap-of-Faith Analysis

Leap-of-faith uses temporal and spatial separation, and to a smaller extent also asymmetric cost wars. A simple model for the uncertainty related to leap-of-faith is that each path where it is used has a probability (say even as high as 0.9) of having a man-in-the-middle attacker. Let us further assume that the total number of different attackers in the network is 2. Then on the first use the uncertainty of the method is 0.9, on the second use $0.9 * 1/2 = 0.45$. In general, on the kth use the the uncertainty is $0.9^k * (1/2)^k$. Here the first component represents the need for the attacker to be present on all communication attempts in order to escape detection of the attack. The second component represents the need for the attacker to be the same on all uses.

6 Previous Work

Much literature exists in all particular areas of weak security discussed above, such as how MIPv6 works without trusted parties. Many weak authentication mechanisms have already been defined and deployed, such as those used by TCP, SSH, or SIP.

However, we are not aware of any work that tries to classify and analyze different types of weak authentication techniques in a general setting. Stajano and Anderson have discussed various techniques that are suitable for one application area that appears suitable for weak authentication techniques, Ad Hoc wireless networks [19]. Gehrmann et al have extended this work towards different kinds of devices in a personal area network [6].

Anderson has discussed the economic impacts related to security mechanisms [2], but has not taken the step we are taking here to design protocols based on these impacts.

Josang [7] Yahalom [21], Beth [5] and others have studied formal models for understanding trust and uncertainty.

7 Conclusions

We have reviewed weak authentication techniques and discussed their properties. Our main conclusions are:

- Sometimes imperfect security can be good enough for the task at hand, or even provide excellent security.
- The cost-benefit curve is not linear. Some basic techniques can provide a significant advantage with little or no cost. An example of this is the TCP SYN cookies method [4]. These techniques can typically used all the time, and stronger security can be used where necessary or possible.
- Results of uncertainty, probability, and economic impact analysis are often surprising.
- Protocols should be designed based on the understanding of the issues relating to uncertainty and economic impacts, in a given application setting.
- Some techniques may lead to the "tragedy of the commons" situation. This can be avoided if standards organizations can mandate certain techniques as a part of a protocol specification.

Further work is needed at least in relating weak authentication and its models to the notion of trust. We also need to create formal models to describe trust, uncertainty, economic impacts, and a view to the whole system. Individual weak authentication methods should be studied further and new ones need to be developed for new applications. The role of multiple contacts, user confirmation, and transitive and combined methods needs more work. The continuum between no user confirmation and cryptographically strong user confirmation is also of particular interest.

References

1. Frank Stajano and Ross Anderson. The resurrecting duckling: What next? In *8th International Workshop on Security Protocols*, Cambridge, UK, 2000.
2. Ross Anderson. Why information security is hard - an economic perspective. In *Proceedings of the 17th Annual Computer Security Applications Conference*, December 2001.
3. Jari Arkko, Tuomas Aura, James Kempf, Vesa-Matti Mantyla, Pekka Nikander, and Michael Roe. Securing IPv6 Neighbour Discovery. unpublished manuscript, submitted for publication, 2002.
4. D. J. Bernstein. Syn flooding. http://cr.yp.to/syncookies/idea, 1996.
5. Thomas Beth, Malte Borcherding, and Birgit Klein. Valuation of trust in open networks. In *Third European Symposium on Research in Computer Security (ESORICS 94)*, 1994.
6. Christian Gehrmann, Kaisa Nyberg, and Chris J. Mitchell. The personal CA - PKI for a Personal Area Network. IST Mobile Summit 2002, 2002.
7. Audun Josang. Trust-based decision making for electronic transactions. In L. Yngstrom and T. Svensson, editors, *Proceedings of the Fourth Nordic Workshop on Secure IT Systems (NORDSEC'99)*, Stockholm University Report 99–005, 1999.

8. W. F. Lloyd. Two lectures on the checks to population. Oxford University Press, 1833.
9. Robert Moskowitz. Host Identity Payload and Protocol. Internet Draft draft-moskowitz-hip-05.txt (Work In Progress), IETF, November 2001.
10. Robert Moskowitz. Host identity payload architecture. Internet Draft (expired) http://klovia.htt-consult.com/draft-moskowitz-hip-arch-02.txt (Work In Progress), IETF, February 2001.
11. Robert Moskowitz. Host Identity Protocol implementation. Internet Draft (expired) http://klovia.htt-consult.com/draft-moskowitz-hip-impl-01.txt (Work In Progress), IETF, February 2001.
12. Pekka Nikander. Denial-of-service, address ownership, and early authentication in the IPv6 world, April 2001.
13. Pekka Nikander and Charlie Perkins. Binding authentication key establishment protocol for Mobile IPv6. Internet Draft draft-perkins-bake-01.txt (Work In Progress), IETF, July 2001.
14. Erik Nordmark. Securing MIPv6 BUs using return routability (BU3WAY). Internet Draft draft-nordmark-mobileip-bu3way-00.txt (Work In Progress), IETF, November 2001.
15. Greg O'Shea and Michael Roe. Child-proof authentication for MIPv6 (CAM). *Computer Communications Review*, April 2001.
16. M. Richardson, D. Redelmeier, and H. Spencer. A method for doing opportunistic encryption with IKE, October 2001.
17. Michael Roe, Greg O'Shea, Tuomas Aura, and Jari Arkko. Authentication of Mobile IPv6 binding updates and acknowledgments. Internet Draft draft-roe-mobileip-updateauth-02.txt (Work In Progress), IETF, February 2002.
18. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. Internet Draft draft-ietf-sip-rfc2543bis-09.txt (Work In Progress), IETF, February 2002.
19. Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *7th International Workshop on Security Protocols*, Cambridge, UK, 1999.
20. Hal R. Varian. Managing online security risks. *The New York Times*, June 2000.
21. R. Yahalom, B. Klein, and Th. Beth. Trust relationships in secure systems - a distributed authentication perspective. In *Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy*, pages 150–164, 1993.
22. T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen. SSH protocol architecture. Internet Draft draft-ietf-secsh-architecture-12.txt (Work In Progress), IETF, January 2002.

Weak Authentication (Transcript of Discussion)

Jari Arkko

Ericsson NomadicLab

John Ioannidis: Your analysis of the cost of mounting an attack, and how weak authentication increases this cost, fails to take into account the expected payoff for the attacker. If only one in a million is interesting, but the information to be garnered is ten million times as interesting as the average persons, then it's worth mounting the attack.

Reply: I agree.

Ross Anderson: This isn't just an issue of authentication; the argument for encrypting traffic on the net, for example, has to do with mechanisms like this. It is always assumed that the spooks are prepared to spend money and take risks in penetrating an end system, and have the techniques for doing that. The question is, how do you know which end system to penetrate and how much money to spend? Normally it's only after observing traffic to and from a particular end system that such a decision can be taken. So it's not just use of weak authentication, it's use of even what are considered to be strong mechanisms, such as encryption. Even if we had a PKI, and everybody could send encrypted emails at some low cost, the reason for doing that still carries with it substantial externalities.

Reply: Yes.

Matt Blaze: One question about the economic analysis is that you've looked at the effect on a network globally if you raise the cost of opportunistic attacking by doing opportunistic encryption, but that doesn't tell anybody why they should themselves use it. It seems that if I have an application that requires security I'm much more likely to demand strong authentication, but if I don't perceive a need for security then I don't see any benefit to paying *any* cost for any kind of authentication. So, while this seems to make the case very well for a common good, there's a tragedy of the commons here if there's something everyone could do but doesn't benefit from directly. We get this large global benefit, but there's no real economic model that dictates that any individual should participate. Is there a good answer to this in this case?

Reply: One potential answer is that there are still parties that actually force people to do things, for instance if they design protocols or enforce standards.

Angelos Keromytis: A question I have is whether a system that employs weak authentication can actually be used as an enabling technology for an attacker. Think of the case of opportunistic IPsec, for example, where the authentication doesn't just affect the secure communication links, it also affects the routing in some sense, because what we're doing is establishing pipes and tunnels. An attacker who can predict a future communication between two parties and who is not on the direct communication path between those two parties,

can simply do that weak authentication with either of the two parties. Because of the way IPsec works, he would be able to examine all traffic between the two parties, whereas he couldn't if no authentication at all was used and plain unencrypted unauthenticated traffic was routed over the Internet.

John Ioannidis: So you don't have a man in the middle, you have a man at the edge.

Reply: Right, and it is usually true in this opportunistic IPsec case that you have a DNS that tells you the tunnel end point address so you travel there.

Larry Paulson: I've had a bizarre idea. In terms of mobile phones it's very rare for somebody to ring up a complete stranger, normally they know each other, and they authenticate each other by voice. They know very much who the other person is. The difficulty is to connect that authentication in the heads, with the digital things inside their phones. What if one could imagine that kind of voice based authentication in which the phone displays on the screen "now say the following sentence to the other person", so you say a sentence which is received at the other end. When they don't send it back you extract the key from that. Now you can't do a man in the middle attack because the man in the middle would have to actually say something.

Matt Blaze: AT&T did something like that, they printed a hash of the session key on the device which you could read to each other.

John Ioannidis: On the other hand, many of my phone calls are not to people I really know, but to services I require. The person answering the phone is some droid whose voice I've never heard before, and whose name isn't even real because the names they use are internal code names.

Victoria Stavridou: I have a question about the computational tradeoffs if you're going to make situations where computational power is really limited, where any communication at all is costly. For example devices like the motes[1] that the people at Berkeley are building. Is that part of your plan in all this?

Reply: That's an interesting thought. Usually the trouble with these small devices is that while you might be limited to doing small computations, typically the attackers have better hardware and are not so limited.

Richard Clayton: I think your economics has to put the cost of the attacker being found out as well as the cost of them mounting the attack. I send e-mail back and forth on the basis that, if anybody's running a man in the middle, first of all it's going to cost them a great deal of money to keep on moving that around, and secondly if I ever catch them doing it then I've got a big win in terms of demonstrating that somebody's been attacking me. But equally was it inside China, one might well imagine that there's a man in the middle looking at absolutely everything and therefore the cost of the Chinese government being found out is very low. They'd almost like everybody to know they were found out so they seem like oppressed people even more.

Reply: Yes, and the governments are in the centre of the networks so...

[1] "Emerging Challenges: Mobile Networking for Smart Dust", J.M. Kahn, R.H. Katz and K.S.J. Pister, J. of Commun. And Networks, Vol. 2, No. 3, September 2000.

John Ioannidis: The cost benefit curve in security is not linear, you don't get twice as much security by spending twice as much money, you might get a lot more security or just a little bit more security. There is a school of thought that says "putting in encryption is cheap so we should be doing weak authentication all the time", and then there are practical aspects of actual software deployed where, in practice the actual software that's out there is so poor that I never send PGP mail because it's not worth my time. There are many dimensions to the economics, it's not just how much CPU power it's going to cost you to do the weak encryption or whether you can do stronger authentication if you had more time or more money.

Reply: I think in practice the things that we can actually deploy are a subset, and are going to be protocols executed mandatorily by the machines, so that people don't have to be making PGP attachments or anything like that.

Matt Blaze: Certainly in applications where security is not perceived as a central service, and new devices are being designed, there is an enormous, almost irrational, pressure to design protocols that can be implemented with the absolute cheapest hardware available. If you say this thing needs to do exponentiations, there are pressures that will try very hard to eliminate it if there isn't a good business case for including it.

Reply: Right, so let's throw out RSA and we'll do a new game here.

Is Entity Authentication Necessary?

Chris J. Mitchell and Paulo S. Pagliusi

Information Security Group, Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
{c.mitchell,p.s.pagliusi}@rhul.ac.uk

Abstract. Conventionally, mutual entity authentication is seen as the necessary precursor to the establishment of a secure connection. However, there exist examples of cases where authentication is not needed. The purpose of this paper is to consider this proposition, illustrated by case studies, and to use the findings of this investigation as input for the design of authentication protocols suitable for use in future Internet access environments supporting ubiquitous mobility.

1 Introduction

In the context of secure communications, mutual entity authentication is very commonly seen as the necessary precursor to the establishment of a secure connection. However, there do exist examples of cases where mutual authentication is not necessary, and, indeed, may impose unnecessary overheads on session establishment. The purpose of this paper is to consider this proposition, using case studies as the basis for this discussion. In these case studies we consider the protocols used in the GSM (Global System for Mobile Communications[1]) and 3GPP (3rd Generation Partnership Project[2]) mobile telecommunications systems.

The main application context of these discussions covers the case where there are three entities involved in the authentication exchange: a mobile user, a local AAA (Authentication, Authorisation and Accounting) server, and a remote (home) AAA server. That is, the mobile user wishes to set up some kind of secure link with a 'local' network (with its own AAA server), and the mobile user has a long term cryptographic relationship, typically backed up by some kind of contractual and payment arrangement, with a remote (home) network and AA server. This 'roaming user' model is not only becoming an increasingly common model for Internet access, but it is also fundamental to understanding the air interface security system for present day mobile telecommunications networks (e.g. GSM and 3GPP).

The purpose of this paper is not so much to talk about GSM and 3GPP, but to consider what more general lessons can be drawn regarding future protocol design. In particular, how best should authentication and/or access security be

[1] http://www.gsmworld.com
[2] http://www.3gpp.org

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 20–33, 2004.

designed in the scenario where a mobile user wishes to access the Internet via a multiplicity of different network types? The recently inaugurated IETF PANA (Protocol for carrying Authentication for Network Access[3]) work will provide a general framework for the exchange of authentication messages in this mobile scenario, but will not address the question of exactly how access security should operate. Other relevant work includes the ongoing IST-Shaman project[4].

2 Entity Authentication and Key Establishment

Before proceeding we need to establish some terminology. We use definitions from the Handbook of Applied Cryptography (HAC), [1], and, where relevant, we indicate the relevant section number from the HAC in brackets after the definition.

Entity authentication is the process whereby one party is assured of the identity of a second party involved in a protocol, and that the second has actually participated (10.1.1). Either one or both parties may corroborate their identities to each other, providing, respectively, *unilateral* or *mutual* authentication (10.1.2).

We are particularly concerned here with the case where a protocol simultaneously provides entity authentication (unilateral or mutual) and session key establishment, where this session key (or keys) is used to protect data subsequently transferred. *Key establishment* is a process or protocol whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use (12.2.1). *Key authentication* (sometimes also called *implicit key authentication*) is the property whereby one party is assured that no other party aside from a specifically identified second party (and possibly additional identified trusted parties) may gain access to a particular secret key (12.2.1). *Key confirmation* is the property whereby one party is assured that a second party actually has possession of a particular secret key (12.2.1). *Explicit key authentication* is the property obtained when both (implicit) key authentication and key confirmation hold (12.2.1).

A further property, desirable in some practical applications but not discussed in [1], is *key freshness*. By this we mean the property that the party to a key establishment process knows that the key is a 'new' key. In particular, the party should have evidence that the messages received during the protocol by which the key has been established are 'fresh' messages, i.e. they are not replays of 'old' messages from a previous instance of the protocol.

To see why this property is necessary in addition to implicit or explicit key authentication, consider the following very simple one-pass key establishment protocol. In this protocol we suppose that A and B share a long term secret key K. Entity A chooses a session key K_s and sends it to B in the following message:

$$e_K(K_s||I_B)$$

[3] `http://www.ietf.org/html.charters/pana-charter.html`
[4] `http://www.ist-shaman.org`

where $e_K(X)$ denotes the encryption of data string X using key K, I_B is an identifier for party B and —— denotes concatenation of data items. Note that we suppose here that the encryption algorithm also provides message integrity and origin authentication (e.g. by additionally computing a Message Authentication Code (MAC) using a variant of K — see, for example, [1]).

This protocol clearly provides (implicit) key authentication to B, given that we assume that K is known only to A and B. It also provides key confirmation to B, since the inclusion of I_B means that the message originates from A, and hence A must know K_s. However it should be clear that the protocol does not provide key freshness, since B has no way of telling whether the message has just been generated by A, or is a replay of a message sent by A at any time since K was first established. Of course, this lack of key freshness can easily be rectified by including a time stamp or sequence number within the scope of the encrypted message.

The absence of key freshness would enable an interceptor to force B to keep re-using an 'old' session key, which might have been compromised. It would therefore seem reasonable to make key freshness a requirement for most applications of key establishment protocols. In fact it turns out that the absence of key freshness is a possible source of weakness in the GSM protocol, as we discuss below.

To conclude this discussion, we note that the two critically important properties for most key establishment protocols would appear to be (implicit) key authentication and key freshness. *Explicit* key authentication is not always so important, and is, in any case, achieved once a party receives evidence of use of a key.

3 Case Study I: GSM

We start by considering the GSM air interface security features. For a more detailed discussion of these features see, for example, [2] or [3]. Note that we are concerned here exclusively with the protocol design for GSM, and not with the security of the algorithms used; for a summary of the current security status of the GSM algorithms see, for example, [2].

3.1 Outline of Scheme

The GSM air interface authentication protocol, i.e. the security protocol used across the wireless path between mobile and network, takes place between a mobile telephone (actually the Subscriber Identity Module (SIM) within the telephone) and the network with which it is currently registered, i.e. the network which the mobile is using to make and receive calls. This is performed with the assistance of the mobile user's 'home network', which originally supplied the SIM. These three entities respectively fit the roles of mobile user, local AAA server, and home AAA server described above.

The SIM and the home network share a long term secret key, K_i. When a mobile user first registers with a new network, this network approaches the home network of the user to request the information necessary to perform the air interface authentication protocol. This information is provided in the form of 'triplets', (N, R, K_c), where N is a random challenge (or nonce), R is the expected response to this challenge, and K_c is a secret session key, to be used to encrypt voice data exchanged between the mobile and the visited network[5]. Both R and K_c are computed as functions of N and the long-term secret key K_i.

To conduct the air interface authentication protocol, the visited network sends the mobile the challenge N. The mobile uses its stored value of K_i to compute R and K_c, and sends R back to the network. The network compares the received value of R with the value in the triple provided by the home network, and if the two agree then the mobile is deemed to have been authenticated. If encryption of the air interface link is required (this decision is made by the network) then the key K_c is used for this purpose.

3.2 Properties of Scheme

We provide only a brief analysis of the scheme. For a more detailed analysis see, for example, [4] or [2].

First observe that the long term secret key K_i is not passed to the visited network. This to some degree limits the trust required in the visited network. Also, since the computation of R and K_c from N and K_i is only performed by the SIM and the home network, the cryptographic algorithms used can be network specific.

The protocol provides unilateral authentication of the mobile to the local network. The protocol also provides (implicit) key authentication to the mobile user, since the key K_c is computed using a long term secret known only to the SIM and the home network. However, the protocol does not provide key freshness to the mobile, since the mobile has no way of determining whether N is a 'fresh' challenge. Indeed, when the local network has run out of triplets and cannot, for some reason get access to any more triplets from the home network, it is allowed to re-use them.

3.3 Analysis

It has been stated on many occasions that the fact that GSM only provides unilateral authentication, i.e. of mobile to base station but not vice versa, is to blame for certain known security weaknesses with GSM. These weaknesses have been widely documented — see, for example, [1,4]. They include the possibility of a false base station interposing itself between the genuine base station and the mobile, and using this to monitor all traffic passing to and from the mobile.

[5] N and R are commonly referred to as 'RAND' and 'XRES'/'SRES' respectively

However, detailed analysis of these GSM weaknesses reveals that adding base station authentication to GSM will not necessarily prevent these problems. A false base station could still insert itself between the mobile and genuine base station *after* a mutual authentication process, since there is no integrity protection for the exchanged traffic. Note also that adding routine integrity protection to the air interface link would not really be viable, since the air interface link is subject to a high level of errors. Deleting all traffic containing errors is not an acceptable strategy, since digitised speech can 'survive' a modest number of transmission errors; that is, enforcing integrity protection would potentially transform a poor quality but usable speech channel into no channel at all.

In fact, the interposition attack would normally be made pointless by routine GSM encryption. However, problems arise because the base station instructs the mobile whether or not to use encryption. A 'false' base station can therefore tell the mobile not to employ encryption, which gives rise to one of the main causes of weakness in the GSM scheme.

One solution to this problem is to provide integrity protection for certain security-critical signalling messages sent across the air interface, notably including the 'cipher enable/disable' messages, and this is precisely the solution adopted in 3GPP (see below). This integrity protection can be based on the session key (or keys) established during the air interface authentication protocol. This leads us to a second problem with GSM, namely the lack of key freshness for this protocol already mentioned above.

This lack of key freshness means that if a malicious third party ever obtains a valid triplet (N, R, K_c) for a mobile, then this can be used to launch an effective false base station attack without suppressing encryption on the radio path. This is discussed in more detail in [4].

4 Case Study II: 3GPP

We next consider a much more recent air interface authentication protocol, designed as part of the 3GPP system. This protocol, specified in [5], has very similar objectives to the GSM protocol and, like GSM, is based on secret key cryptography. In fact, the design of the 3GPP protocol is closely based on the GSM scheme, taking into account the known problems with the GSM protocol.

4.1 An Enhanced GSM Protocol

Before giving a description of how 3GPP operates, we give an example of an 'enhanced' version of GSM. This example is not a serious proposal for adoption, but is intended to help explain the details of the 3GPP design, and what could go wrong if some of the features were not present.

Suppose that the mobile user's 'User Services Identity Module' (USIM), i.e. the successor to the GSM SIM, is equipped with a sequence number S (initially set to zero) as well as a long term shared secret K_i. Instead of providing triplets to the visited network, the home network provides 'quadruplets' (N, R, K_c, S),

where N is as before, R and K_c are as before except they are a function of S (as well as N and K_i), and S is a sequence number. The home AAA server keeps a record of S for each mobile user, and generates quadruplets with monotonically increasing sequence numbers. (Note that the same effect can be achieved without keeping a database of sequence numbers - see, for example, [6]).

To authenticate the mobile, the visited network sends the challenge and serial number to the mobile (i.e. N and S). As long as S is larger than any previously received sequence number, the mobile accepts it, updates its stored sequence number, and computes the response R (and the session key K_c) as a function of N, S and K_i. This revised protocol now provides key freshness, since the mobile can check that S is fresh and moreover the session key is a function of S.

To address the other problem with GSM discussed above, a second session key could also be derived at the same time, and used to protect the integrity of security-critical signalling messages. By this means the major weaknesses of GSM could be addressed.

However, there is one major problem with this solution. That is, there is a trivial and fatally serious denial of service (DoS) attack. An attacker can simply send a very large (maximal) sequence number S to a mobile, along with an arbitrary challenge N. The mobile will set its stored sequence number to this very large value, and will thereafter never accept any more challenges from the genuine base station. The existence of this DoS attack motivates the slightly more complex design of the actual 3GPP protocol, which we now describe.

4.2 Outline of 3GPP Authentication

As in the 'enhanced GSM' protocol, the mobile user's USIM is equipped with a sequence number S (initially set to zero) as well as a long term shared secret K_i. Instead of providing triplets to the visited network, the home network provides '6-tuples' (N, R, K_c, K_a, S, M), where N, R and K_c are as in GSM, K_a is a second session key derived like K_c as a function of N and K_i (the long term secret key), S is a sequence number, and M is a MAC computed with data input a concatenation of N and S, and with secret key input K_i, the long term secret key[6]. As above, the home AAA server keeps a record of S for each mobile user, and generates 6-tuples with monotonically increasing sequence numbers.

To authenticate the mobile, the visited network sends the challenge, serial number and MAC to the mobile (i.e. N, S and M). The mobile first checks the MAC. If the MAC verifies correctly, and as long as S is larger than any previously received sequence number, the mobile accepts it, updates its stored sequence number, and computes the response R (and the session keys K_c and K_a) as a function of N and K_i.

[6] N, R, K_c, K_a, S and M are commonly referred to as 'RAND', 'XRES'/'SRES', 'CK', 'IK', 'SQN' and 'MAC' respectively, and the concatenation of S and M is commonly referred to as 'AUTN'. Note also that the above is a slightly simplified description of 3GPP authentication — in the actual scheme the sequence number S is sent encrypted to prevent it revealing the identity of the mobile user

The 3GPP protocol provides key freshness, since the mobile can check that S is fresh and moreover the challenge N is 'bound' to S by the MAC M. Thus, since the session keys are functions of N, they are also guaranteed to be fresh.

The session key K_c is used to encrypt data sent across the channel (just as in GSM) and K_a is used in parallel to protect the integrity of security-critical signalling messages. By this means the major weaknesses of GSM are addressed.

4.3 Properties of the 3GPP Protocol

The 3GPP protocol clearly does not suffer from the DoS attack to which the 'enhanced GSM' protocol is prone, because of the presence of the MAC M. The 'false base station in the middle' problems are removed by providing data integrity and data protection for security-critical signalling messages from the base station to the mobile, including the message to enable encryption (this latter message is mandatory, so simply deleting it will not be an effective attack).

It is also worth noting that providing integrity protection for signalling messages without simultaneously providing key freshness would not be sufficient to deal with attacks arising from compromise of old session keys.

Finally note that it is the provision of key freshness combined with signalling integrity, not local network authentication, that prevents the false base station attacks to which GSM is prone. Nevertheless, and unlike GSM, mutual authentication is provided in 3GPP. This is because the inclusion of a MAC in the message sent to the mobile from the network enables the mobile to authenticate the source of the message, and the sequence number enables the message to be checked for freshness.

This would appear to undermine the main thesis of this paper, i.e. that mutual entity authentication is not always necessary. However we claim that the provision of network authentication to the mobile in 3GPP is essentially an accident. That is, it is provided only as an accidental by-product of the provision of other necessary security services. To support, this claim, first observe that the 'enhanced GSM' protocol in Section 4.1 does potentially deal with the main GSM problems, as long as it is used in conjunction with signalling integrity - the only problem with this protocol is the 'new' issue of a serious DoS attack. Secondly, in the next section we present a protocol which meets all the requirements of the 3GPP scenario without providing authentication of the network to the mobile and which is not prone to a DoS attack.

4.4 Other Remarks

In fact there are additional advantages of the 3GPP protocol structure (as compared to the 'enhanced GSM' protocol), specific to the 3GPP operational environment. One potential advantage of sending the MAC M (apart from DoS prevention) is that it avoids the need to use S in calculating the session keys. This might be important for GSM/3G interoperation. In the 3GPP protocol, where the session keys are based on N and the long term secret K_i only, the GSM authentication triplet can be derived from the 3G authentication 6-tuple

by simply ignoring S and M, and using a simple conversion function to derive a 64-bit GSM K_c from the two 128-bit keys K_c and K_a. This is important because dual mode mobiles will have to work in GSM networks which cannot handle 3G authentication 6-tuples.

One other reason for the MAC M is that it can be used to protect the 'AMF' field. AMF is an undefined data string concatenated with S which may be used for operator specific commands to the card.

5 Case Study III: 3GPP-Like Protocols

We now briefly consider another 3GPP-like protocol. The main motivation for presenting this protocol is to show that the distinction between the need for specific properties for an established key and mutual authentication is real. In particular, because 3GPP provides mutual authentication, the suspicion might arise that the only way in which key freshness and key authentication can sensibly be obtained is to use a mutual entity authentication protocol.

The mobile user's USIM is here assumed to possess a clock synchronised to the clock of the network, as well as a long term shared secret K_i. Instead of providing 6-tuples to the visited network, the home network provides '5-tuples' (N, R, K_c, K_a, T), where N and R are as in 3GPP, K_a and K_c are derived as a function of N, T and K_i (the long term secret key), and T is a timestamp. This scheme requires 5-tuples to be used within a short period of their generation, as the mobile will check that T is current.

To authenticate the mobile, the visited network sends the challenge and timestamp to the mobile (i.e. N and T). The mobile first checks the timestamp to see if is within the 'window of acceptance'. If so, the mobile accepts it and computes the response R (and the session keys K_c and K_a) as a function of N, T and K_i.

The 3GPP protocol provides key freshness, since the mobile can check that T is fresh and moreover the session keys are functions of T. Just as in 3GPP, the session key K_c is used to encrypt data sent across the channel (just as in GSM) and K_a is used in parallel to protect the integrity of security-critical signalling messages. Finally, this scheme is not prone to the DoS attack, since the mobile will not reset its clock.

However, the protocol clearly does not enable the mobile to authenticate the network. Thus clearly entity authentication is not always required in order to establish key freshness and key authentication. However, this protocol is not a serious candidate for use in the 3GPP scenario, since assuming synchronised clocks (and the accompanying management overhead) is not reasonable in this environment.

6 Future Systems

One issue which we have not examined in detail here is the fact that, in any mobile user scenario, there would appear to be a need to delegate some ac-

cess security functions from the 'home' AAA server to the visited network AAA functionality. The question remains open as to how best this should be done, especially bearing in mind possible anonymity requirements and trust issues. (Anonymity is an issue partially dealt with by GSM, and more thoroughly provided by 3GPP, but is beyond the scope of this paper).

Exactly what security services are really required for which protocols, in particular is mutual authentication a genuine requirement, and how much of the provision of these services should be delegated to the visited network? Whilst 3GPP might provide a model for a solution based on symmetric cryptography, how should we solve the same types of problem using asymmetric cryptographic techniques? This latter question appears to be of importance, because of the advantages to be gained from the use of public key techniques in scenarios where the number of entities proliferates, and there is no single model for establishing bilateral trust relationships, e.g. as in the PANA workgroup scenarios.

7 Concluding Remarks

By considering a number of example protocols, we have provided evidence for the view that entity authentication is not always an essential precursor for the establishment of secure communications. In the case of GSM, it is often claimed that the lack of mutual entity authentication is the source of certain well known problems. However, it is clear that not only is this not the case, but also mutual authentication on its own will not solve the problems.

We have further argued that, in the typical case, the most important issue is to ensure that the properties of (implicit) key authentication and key freshness are provided for any established session keys. These session keys can be used to protect the integrity of security-sensitive data exchanged during the session, thereby preventing 'man in the middle' attacks.

Of course, key freshness can be obtained 'for free' if the key establishment process is embedded within a mutual entity authentication protocol. Indeed, it is this fact that may perhaps be responsible for the fact that the issue of key freshness is not widely discussed in the literature examining key establishment protocols (notably it is omitted from the discussion in [1]). However, the importance of the key freshness property means that, if protocols are designed which do not provide mutual authentication, then it is vital that the provision of key freshness is carefully checked.

Acknowledgements. The authors would like to acknowledge the many helpful insights and corrections provided by Peter Howard, Keith Martin, Kenny Paterson and Scarlet Schwiderski-Grosche. The authors would also like to thank all the participants in the IST Shaman project for their advice and encouragement.

References

1. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
2. Walker, M., Wright, T.: Security. In Hillebrand, F., ed.: GSM and UMTS: The creation of global mobile communication. John Wiley & Sons (2002) 385–406
3. Sutton, R.: Secure communications: Applications and management. John Wiley & Sons (2002)
4. Mitchell, C.: The security of the GSM air interface protocol. Technical Report RHUL-MA-2001-3, Mathematics Department, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK (2001) Available at `http://www.ma.rhul.ac.uk/techreports`.
5. Third Generation Partnership Project: 3GPP TS 33.102: Security Architecture. V3.11.0 edn. (2002) Technical Specification Group Services and System Aspects, 3G Security, Valbonne, France.
6. Mitchell, C.: Making serial number based authentication robust against loss of state. ACM Operating Systems Review **34** (2000) 56–59

Is Entity Authentication Necessary? (Transcript of Discussion)

Chris J. Mitchell

Information Security Group, University of London

Ross Anderson: In my book[1], in the chapter on telecoms security, I point out that GSM security was designed to protect the phone companies, to do no serious harm to the spook agencies, and to hell with the users. Your comments about 3GPP and GSM, that it might enable one user to generate a large bill for another, basically digs this up in spades. As far as the average user is concerned, being listened to by funny people in the West Country is not an issue, unless you're suffering from mental illness. What is an issue is that you might get a very, very large bill as a result of either some hacker or, perhaps more to the point, somebody in a phone company corruptly entering into an agreement with a sex line number operator to generate large numbers of minutes from your number. This is a serious problem; it does cause, globally perhaps nine figures of fraud, an awful lot of grief and embarrassment to individuals, and it is very, very difficult to get the thing set right by the phone company. So I certainly hope that in your SHAMAN project you will bear in mind that the customer, the owner of the equipment, is also a participant in the protocol, not just the customer in the sense of the phone company that writes the protocol.

From the users point of view, this is the issue. Confidentiality isn't an issue. Nobody cares. To the first order of approximation, there's no phone tapping in the world because nobody's interested. But a lot of people are interested in stealing your money.

Matt Blaze: There is still a cost to the phone company: the customer service cost in dealing with people complaining about their bill can be very high.

Ross Anderson: Well that's why you've got phone robots.

Matt Blaze: I just want to suggest an alternate view. You questioned whether we need mutual authentication at all, and you cited complexity as an argument against it. I'm going to take issue with that just from the protocol engineering point of view. It seems to me that if you take out mutual authentication from a security protocol, what you're ultimately going to do is confuse the clients of that protocol - the higher level services that are using it - and force them to implement authentication services on top of that with a very unclear model below it. And invariably this will be done without security protocol designers doing it. But I think from the point of view of complexity, if we consider complexity very broadly and not strictly in terms of computational cycles, we may be much better off having security services that are very, very strong; security services that can be viewed as a black box in as broad a sense as pos-

[1] Ross J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, John Wiley & Sons 2001 - ISBN: 0471389226.

sible. That would avoid requiring higher level services to implement things for themselves.

Reply: I take your point. My implicit model is that attacks are going to be quite rare, especially if there's not much to gain by making an attack. You can, as it were, complete the protocol, but as you don't get the keys you can't actually conduct a call, so nobody would even bother.

Bruce Christianson: Very often there is a kind of implicit delegation. The lower level protocol is going to generate some sort of shared key which it is then going to hand off to a higher level protocol, which is going to use it to secure a screen session or something at a much higher level. There are two observations to make here. The first observation is that very often the low level designer has forgotten that the high level design may do things like reveal the key before the low level protocol is even finished, and there are some wonderful howlers there. But secondly, the change in the abstraction boundary is often accompanied by a shift in who the other participant is. 'Bob' shifts from being a piece of hardware owned by X, to a piece of software acting as a proxy for Y and, very often, the mechanism for the handing off doesn't reflect that. But I agree with you that the actual nuts and bolts should be done at a lower level because that's where the appropriate physical resources are.

Pekka Nikander: When we're doing this kind of delegation I think it is very much about the underlying trust model. But more importantly I think you should really concentrate on the external effects that the protocols have, like generating bills and so on. Although you say you are not doing authentication, I think you are actually. You're not doing it between the base stations and the mobile, you're doing it between the base station, the mobile and the home agent. You can do this because of your trust model and because of your expectation about the external effects.

Reply: Yes, I'm not sure you would disagree with this, but another way of looking at what I'm trying to say is there's often this model of "we must do an entity authentication, it's a really good top-notch thing" at the beginning, and then subsequently we use the established session keys to protect the session. I guess I'm suggesting regarding these as one thing rather than two. You'll get some of the authentication you need when you conduct the session, because once you use the session key, you are authenticating yourself. If you've got implicit key authentication up to that point, then as soon as you start using the key, you prove that there's liveness there. The other guy knows exactly who that person must be because they're using the key which only they could know. There are some dangers, because now you've got to engineer the whole shooting match (the authentication protocol and the actual session) in a secure way. I take your point, but maybe there are economies of scale in trying to regard them as one, rather than two separate things.

Wenbo Mao: I heard you say that mobile users don't care about confidentiality. Billing is more important. Why is this?

Dieter Gollmann: Just sit on any commuter train, and you can eavesdrop on any number of people with mobile phones.

Reply: One thing you have to bear in mind is that with GSM, and all the mobile systems I'm aware of, the encryption only happens on the radio link between the mobile and the local base station. When your call is routed through the fixed network it's typically unencrypted. So if you want to eavesdrop, it's necessarily not so hard. With a fixed network, which people kind of regard as being roughly equivalent, eavesdropping is again trivial. There's probably a green box outside your house, there certainly is outside mine. It opens with a standard key and inside is the network. If you know what you're doing and you have a couple of crocodile clips you could tap into a telephone call without any difficulty. So people don't expect the telephone network to be terribly secure from the point of view of confidentiality.

John Ioannidis: It's not something they notice right away like a bill with the extra 10 pence of charges, but it is only in their immediate sphere of awareness that there is such a thing as confidentiality. I mean, people still use cordless phones that you can listen in to with a simple radio receiver.

Mike Bond: Can I just add a comment here? When you've got encryption over some radio link it can protect against hobby eavesdroppers. This may sound a fairly pointless set of people to protect against, but to give an example from back in the days of analogue mobiles: company executives would come round, then they'd look at the company and then use their mobiles to talk about the latest takeover or something. Junior staff could tune in with their scanners and get advance information about what the stock prices were going to do. So there you have a spatial location between the attacker and the people being attacked, and so raise the bar for this sort of casual person.

Reply: I guess that is why there is some encryption, but of course it's difficult to know why it was put there. My interpretation is that it was put there to give a level of protection roughly equivalent to that which you'd get from using the wired network, and it probably achieves that.

Larry Paulson: It would be interesting to know what real attacks have occurred rather than just theoretical ones. For example, have there been any false base station attacks, and if so, what damage have they done?

Reply: Certainly false base stations exist.

Ross Anderson: You can buy them (though you are supposed to be a police force in order to do so).

Larry Paulson: So they let you listen in on calls which you had to pay for?

Tuomas Aura: There are good reasons for buying base stations, for example if you want to be your own access provider in your home or in your company.

Ross Anderson: A genuine application would be that you set up a false base station in a concert hall. When somebody phones one of the people listening to your concert they would get a message saying "Dr Jones is listening to Vivaldi sir, is it really important enough for me to send a chap to fetch him?" You could charge them $14 a minute to call the hall.

Matt Blaze: Other than the operator, that is a commercial product. It's not available everywhere, but it's the law in most, not all, countries. There is a company that markets essentially the concert hall base station.

Tuomas Aura: But what they do is they jam the unfortunates.

Matt Blaze: Well the latest generation of products actually registers the phone, so it thinks it's happily talking to a home network.

Ross Anderson: But there is a problem with these meet in the middle attacks. It's what I was talking about earlier. The person who can do the meet in the middle attack, whether directly or through exploiting cryptoanalysis or whatever, puts himself into a position to bill a very large number of minutes to the attacked mobile. And there are all sorts of feature interactions you have to worry about, and potential tricks with conference calls. It seems to me to be horrendously difficult to design a system that supports all the features you get in modern telephone systems, and yet can't be used in an abusive way to rip off customers via the 0900 number system.

John Ioannidis: It's not just ripping off customers, it's also embarrassing them like they did to our former Mayor in New York by charging about $1m to his cell phone.

Pekka Nikander: The underlying reason I think is that the operators trust each other. When you put up a network that doesn't belong to the operators you are breaking the underlying trust.

Ross Anderson: What about when you put operators into the network who are untrustworthy?

Pekka Nikander: So, the basic assumption that they made way back when they were developing the GSM was more or less wrong?

Ross Anderson: It gave about the same amount of trust as there is in the wired network, but that was inadequate from the customers point of view. What we should perhaps see is the redesign of these mechanisms, bearing in mind that ultimately the customer is the user. Although you can rip off the users for a period of years by making them vulnerable to all sorts of scams, in the end there will be a court case which stops that. That can prove horrendously expensive for the phone companies who then have to do lots of reengineering.

Reply: I'll just say that my talk wasn't really about GSM. GSM was an example.

John Ioannidis: Yes, it's the sort of example that tends to bring up a lot of repressed anger, or not so repressed anger. [Laughter]

A Structured Operational Modelling of the Dolev-Yao Threat Model

Wenbo Mao

Hewlett-Packard Laboratories
Filton Road, Stoke Gifford
Bristol BS34 8QZ
United Kingdom
wm@hplb.hpl.hp.com

Abstract. In the areas of computer security and cryptography a standard model for adversaries is the *Dolev-Yao threat model*. In the areas of formal analysis of complex, concurrent, communication and reactive systems, one of the foundations for formal analysis methodologies is a *structured operational semantics (SOS)* for Milner's process algebra *Calculus of Communicating Systems (CCS)*. In this paper we provide a CCS-SOS modelling of the Dolev-Yao threat model. The intuitively appealing modelling indicates a suitability for the well studied formal analysis methodologies based on CCS-SOS being applied to computer security and cryptography.

1 Introduction

In the areas of computer security and cryptography a standard model for adversaries is the *Dolev-Yao threat model* [4]. In the areas of formal analysis of complex, concurrent, communication and reactive systems, one of the foundations for formal analysis methodologies is a *structured operational semantics (SOS)* for Milner's process algebra *Calculus of Communicating Systems (CCS)* [9,10]. In this paper we provide a CCS-SOS modelling of the Dolev-Yao threat model. The intuitively appealing modelling indicates a suitability for the well studied formal analysis methodologies based on CCS-SOS being applied to computer security and cryptography.

The remainder of the paper is organised as follows. In §2 we introduce the Dolev-Yao threat model. In §3 we introduce CCS-SOS. In §4 we introduce a model checker for security protocols named Brutus (which provides us with a syntax for specifying the CCS-SOS modelling of the Dolev-Yao threat model). In §5, we present the CCS-SOS modelling of communication principals' behaviour under the Dolev-Yao threat model. Finally, in §6 we discuss possible further work to be done.

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 34–46, 2004.

2 Vulnerable Environment (The Threat Model of Dolev and Yao)

A large network of computers, devices and resources (for example, Internet) is typically open, which means that a **principal** (or **entity**, **agent**, **user**), which can be a computer, a device, a resource, a service provider, a person or an organisation of these things, can join such a network and start sending and receiving messages to and from other principals across it, without a need of being authorised by a "super" principal. In such an open environment we must anticipate that there are **bad guys** out there who will do all sorts of bad things, not just passively eavesdropping, but also actively altering (maybe using some unknown calculations or methods), forging, duplicating, re-routing, deleting or injecting messages. The injected messages can be malicious and cause a destructive effect to the principals in the receiving end. In the literature of computer security and cryptography such a bad guy is called an (**active**) **attacker** (or **adversary**, **enemy**, **eavesdropper**, **intruder**, etc). In this paper we shall name an attacker **Malice** who is someone who desires to do harm or mischief, and often does so under the masquerade of a different identity, such as Alice. Malice can be an individual, a coalition of a group of attackers, and, as a special case, a legitimate principal in a protocol (an **insider**).

In general, Malice is assumed to be very clever in manipulating communications over the open network. Her manipulation techniques are unpredictable because they are unspecified. Also because Malice can represent a coalition of bad guys, he may simultaneously control a number of network nodes which are geographically far apart. In anticipation of such a powerful adversary over such a vulnerable environment, Dolev and Yao proposed a **threat model** which has been widely accepted as the standard thread model for cryptographic protocols [4]. In that model, Malice has the following characteristics:

a) he can obtain any message passing through the network;
b) he is a legitimate user of the network, and thus in particular can initiate a conversation with any other user;
c) he will have the opportunity to be a receiver to any principal;
d) he can send messages to any principal by impersonating any other principal.

Thus, in the **Dolev-Yao threat model**, any message sent to the network is considered to be sent to Malice for his disposal (according to whatever he is able to compute). Consequently, any message received from the network is treated to have been received from Malice after his disposal. In other words, Malice is considered to have the complete control of the entire network. In fact, it is harmless to just think the open network to be Malice.

However, unless explicitly stated, we do not consider Malice to be *all powerful* in terms of solving computational problems (even in the case of representing a coalition of bad guys and using a large number of computers across the open network in parallel). This means that there are certain things that Malice cannot do:

- Malice cannot guess a random number which is chosen from a sufficiently large space.
- Without the correct secret (or private) key, Malice cannot retrieve plaintext from given ciphertext, and cannot create valid ciphertext from given plaintext, with respect to the perfect encryption algorithm.
- Malice cannot find the private component, i.e., the private key, matching a given public key.

3 Calculus of Communicating System and Its Structured Operational Semantics

The behaviour of a protocol as a finite-state transition system will be described by a labelled transition system (LTS). This is achieved by following a process algebra approach. We will use a subset of a process algebra named CCS to describe protocol participants' transition behaviour and system composition.

CCS stands for *Calculus of Communicating Systems* and is mainly the work of Milner [9,10]. CCS is regarded as the first attempt to apply process model to the analysis of the concurrent communication behaviour of systems. The subset of CCS terms to be used here can be illustrated as follows:

- 0 ("inaction")
  - do nothing; same as "*Stop*" in CSP [6,7];
- aP ("prefix")
  - perform action a and then behave like P; same as "$a \rightarrow P$" in CSP;
- $P + Q$ ("choice")
  - behave like P or Q; similar to "$P \,\Box\, Q$" in CSP;
- fix.$E(x)$ ("fixpoint"; here $E(x)$ is a CCS expression with a variable x)
  - same as "$\mu X.E(x)$" in CSP;
- $P \mid Q$ ("concurrency" and "interleaving")
  - the combination of "$P \parallel Q$" and "$P \,|||\, Q$" in CSP.

These terms have a Structured Operational Semantics (SOS) which is defined in Figure 1. This is a transition relation over the CCS terms. Since we only use a subset of the CCS terms, the SOS transition relation given in Figure 1 is also a cut-down set of CCS's SOS transition relation.

In CCS, a step of communication action between two processes can be described by a "handshake" transition provided that the two processes can perform the same *external* actions but in the *complement* forms (see transition rule "Handshake" in Figure 1). The complement form of an action a is denoted by $\bar{a}$. A pair of complement actions (a, $\bar{a}$) models an action involving signal output and input, respectively. Let $\bar{\mathcal{M}} = \{\bar{m} : m \in \mathcal{M}\}$. The set of actions is

$$Act = \mathcal{M} \cup \bar{\mathcal{M}} \cup \{\tau\}.$$

Act is the set of all *sent* messages (in $\mathcal{M}$) and *received* messages (in $\bar{\mathcal{M}}$), plus a so-called *invisible* (also called *silent*) action τ.

$aP \xrightarrow{a} P$	Guarding
if $P \xrightarrow{a} P'$ then $P + Q \xrightarrow{a} P'$ and $Q + P \xrightarrow{a} P'$	Choice
if $E(\text{fix}.E(x)) \xrightarrow{a} P$ then $\text{fix}.E(x) \xrightarrow{a} P$	Fixpoint
if $P \xrightarrow{a} P'$ and $Q \xrightarrow{\bar{a}} Q'$ then $P\|Q \xrightarrow{\tau} P'\|Q'$ and $Q\|P \xrightarrow{\tau} Q'\|P'$	Handshake
if $P \xrightarrow{\tau} P'$ then $P\|Q \xrightarrow{\tau} P'\|Q$ and $Q\|P \xrightarrow{\tau} Q\|P'$	Interleaving

Fig. 1. Structured Operational Semantic (SOS) Transition Rules

CCS has postulated two SOS transition rules for the invisible action τ. These two rules are named "Handshake" and "Interleaving" in Figure 1. We should provide some explanations on the intuitive postulation of these two rules.

- The rule "Handshake" says the following. If P and Q can communicate in a self-supply-self-consumption manner then their composition should have no interaction with their external environment and hence the interaction between them should be invisible by an external observer. However, we must notice that this sense of "invisibility" is only valid in a "gentleman's environment". We will provide an intuitive model for Malice in that τ is visible to him!
- The rule "Interleaving" says the following. If an action is invisible by a system component then the component should not be aware of the system's state transition progress. Again, this intuition is only valid if the component models an honest part of the system. We will provide an intuitive model for Malice in that he can manipulate an interleaving transition.

CCS has defined various notions of equivalence relation over CCS terms. We shall not introduce them. An axiomatisation of an equivalence (written as equation $=$) with respect to processes transition behaviour suffices for our use and is listed in Figure 2. It is obvious that this axiomatisation is consistent with respect to the SOS transition rules in Figure 1.

4 The Brutus Model

The model checker **Brutus** first appeared in [1]. We describe here its most recent version in Marrero's Ph.D. Thesis [8].

$P \circ P = P$	Absorption for $\circ \in \{+, \mid\}$
$P \circ Q = Q \circ P$	Commutativity for $\circ \in \{+, \mid\}$
$P \circ (Q \circ R) = (P \circ Q) \circ R$	Associativity for $\circ \in \{+, \mid\}$
$P \mid (Q + R) = (P \mid Q) + (P \mid R)$	Distribution of $\mid$ over $+$

Fig. 2. Axioms

When a model checking technique works for security protocols analysis, *system composition* models the communications among the protocol participants, and in particular the protocol participants may include Malice. The original work of Brutus ([1,8]) does not provide a syntactic nor a semantic treatment on system composition. We believe that an explicit description of system composition is important in a model checking approach to the analysis of a communication system. Such a system is a composition of several components who communicate with each other. With system composition properly described, we can spell each system component in a step-wise manner to ensure that each of them is modelled precisely, and thereby the whole system can be built up through composition, also precisely. Therefore, in our description of Brutus we will explicitly use the CCS-SOS for describing system composition.

We will also take a few steps of simplification in order to make our description to be more suitable for the reader who is not specialised in formal methods areas.

4.1 Protocol Messages

- $\mathcal{P}$: set of honest principals' names. The elements in $\mathcal{P}$ are usually represented by capital letters A, B, ..., T standing for Alice, Bob, ..., Trent, respectively.
- $\mathcal{P}_\Omega$: principals names including Ω, the bad guy, i.e., Malice.
- $\mathcal{N}$: the set of nonces, timestamps, sequence numbers. The elements in $\mathcal{N}$ are usually represented by N_A, N_B, ..., standing for nonces generated by the respective principals.
- $\mathcal{K}$: the set of all cryptographic keys. The system uses three functions to associate keys to principals:

$$pubkey : \mathcal{P}_\Omega \mapsto \mathcal{K}$$

$$privkey : \mathcal{P}_\Omega \mapsto \mathcal{K}$$

$$symkey : 2^{\mathcal{P}_\Omega} \mapsto \mathcal{K}$$

So $pubkey(A)$ is A's public key and $privkey(B)$ is B's private key. $2^{\mathcal{P}_\Omega}$ is the power set of $\mathcal{P}_\Omega$; so $symkey(A, B)$ denotes the a symmetric key shared

between A and B. Every key $K \in \mathcal{K}$ has an inverse $K^{-1} \in \mathcal{K}$ such that for all messages M,

$$\{\{M\}_K\}_{K^{-1}} = M.$$

The three key functions have the following inverses:

$$(pubkey(X))^{-1} = privkey(X)$$

$$(privkey(X))^{-1} = pubkey(X)$$

$$(symkey(X))^{-1} = symkey(X)$$

For presentation simplicity we will often use the more conventional abbreviations for the key association functions:

$$pubkey(X) = K_X$$

$$privkey(X) = K_X^{-1}$$

$$symkey(X, Y) = K_{XY}.$$

– $\mathcal{A}$: the set of all atomic messages:

$$\mathcal{A} = \mathcal{P}_\Omega \cup \mathcal{N} \cup \mathcal{K}.$$

– $\mathcal{M}$: the set of all messages over $\mathcal{A}$. This is defined inductively as follows.
  - If $a \in \mathcal{A}$ then $a \in \mathcal{M}$ (any atomic message is a message).
  - If $m_1 \in \mathcal{M}$ and $m_2 \in \mathcal{M}$ then $m_1 \cdot m_2 \in \mathcal{M}$ (two messages can be paired together to form a new message).
  - If $m \in \mathcal{M}$ and $K \in \mathcal{K}$ then $\{m\}_K \in \mathcal{M}$ (a message m can be encrypted with key K to form a new message).

4.2 Information

The notion of information is about the knowledge of a principal. Let I denote an initial set of information of a principal. Then the principal can derive new information from the known ones. We can capture the way for information derivation using a derivation relation "$\vdash$" as follows.

– If $m \in I$ then $I \vdash m$.
– If $I \vdash m_1$ and $I \vdash m_2$ then $I \vdash m_1 \cdot m_2$ (paring).
– If $I \vdash m_1 \cdot m_2$ then $I \vdash m_1$ and $I \vdash m_2$ (projection).
– If $I \vdash m$ and $I \vdash K \in \mathcal{K}$ then $I \vdash \{m\}_K$ (encryption).
– If $I \vdash \{m\}_K$ and $I \vdash K^{-1} \in \mathcal{K}$ then $I \vdash m$ (decryption).

We can understand the difference between messages and information as follows: messages are those which have been specified in a protocol, while information is about a principal's knowledge, it is the set of elements which can be derived by a principal using the knowledge s/he has; these elements need not have been specified in a protocol.

When Malice is trying to defeat the goal of a protocol, he can only use information which he can derive from some initial set of information and some protocol messages which have been sent to the network. Usually, Malice will take some active actions to help his information derivation, for example, by sending information he has in an attempt to obtain oracle services from honest principals.

By the way of information derivation, I is in principle an infinite set. However, in reality, given a protocol, we can always limit I to a finite set of "interesting" information. Moreover, researchers have taken advantage of the fact that there is not need to actually construct I. It suffices to check $m \in I$ for some finite number of messages.

5 CCS-SOS Modelling of the Behaviour of Principals

Now we are ready to model the behaviour of a protocol participant, i.e., a principal. Each principal is modelled by a triple:

$$(X,\ X.I,\ X.P)$$

where

- X: principal name;
- $X.I$: X's information set;
- $X.P$: X's process description.

Here $X.I$ and $X.P$ are state systems. A principal so modelled is called a *local state*. The modelling of a principal local state will be given in a step-wise spelling manner. Only in so doing we can achieve a desired precision.

5.1 Message Sending

Transition $X.P \xrightarrow{m} X.P'$ can occur in the following environment settings:

Principal X :
$X.P = mX.P'$ (current process description for X);
$m \in X.I$ (m is in X's information set);
$X.I' = X.I$ (sending a message does not change one's information);
Bad guy :
$\Omega.I' = \Omega.I \cup \{m\}$; ($m$ is actually sent to the bad guy)
Other principals : no change.

5.2 Message Receiving

Transition $X.P \xrightarrow{\bar{m}} X.P'$ can occur in the following environment settings:

Principal X :
$X.P = \bar{m}X.P'$ (current process description for X);
$X.I' = X.I \cup \{m\}$ (m is added to X's information set);
Bad guy :
$m \in \Omega.I$ (m is actually coming from the bad guy);
Other principals : no change.

5.3 Internal Actions

There can be various internal actions, e.g., nonce generation, decryption, etc.

Transition $X.P \xrightarrow{X(m)} X.P'$ can occur in the following environment settings:

Principal X :
If $X.P = X(m)X.P'$ (current process description for X);
$X.I' = X.I \cup \{m\}$ if m is new due to the internal action $X(m)$ (information derivation);
Bad guy : no change;
Other principals : no change.

5.4 The Behaviour of Malice

Brutus provides no behaviour modelling for the dishonest principal Ω, that is, there is no definition for $\Omega.P$. This is due to the following consideration: "Ω's behaviour is unknown".

However, according to the Dolev and Yao threat model for open computing and communications network [4], we do know the following typical behaviour of Ω. It can

- receive any cleartext message sent to the network; maybe block the message, or maybe modify it and
- send out any message it possesses and sometimes do so by impersonating any other principals.

The following recursive process models this behaviour intuitively:

$$\text{fix.}(\bar{m}_1 X + m_2 X + \tau X) \tag{1}$$

where m_1 is any protocol message sent by an honest principal, and m_2 is any *information* that Ω has in possession. We should notice two points in this modelling: (1) any message sent by an honest principal is considered to be sent to Ω, and hence is modelled here to have been received by Ω; (2) any *information* (not only message) possessed by Ω may be sent by him at any time, or may eventually be sent; when Ω chooses not to send it for the time being, he may either receive any message sent by other (honest principals, or stay idling, as modelled here with a choice of τ-transition.

Applying the transition rule "Fixpoint", the CCS fixpoint expression in (1) can be unfolded into the following recursive definition:

$$\Omega \stackrel{\text{def}}{=} \bar{m}_1 \Omega + m_2 \Omega + \tau \Omega. \tag{2}$$

For simplicity in presentation we have used Ω (a principal's name) to represent the process of Ω. Doing so can cause no confusion. The recursive definition for Ω provides an intuitive modelling of Malice's unchangeable capability to interact with the network. In particular, it models the fact that Ω is capable of observing the "invisible" action.

5.5 Protocol Behaviour as System Composition of Local States

Having spelt out, in a step-wise manner, all the local states for protocol participants and for Malice, the *global* behaviour of a protocol will be the system composition of all local states. This can be achieved by applying the operation "|". There exists a tool named "Edinburgh Concurrency Workbench" (CWB) [2] which can process the composition mechanically.

Given a protocol, it is obvious that each legitimate principal can be modelled by a finite transition system. If we limit $\Omega.I$ to be a finite set of "interesting" information, then the process Ω defined in (2) is a finite-state transition system. Thus, the global state system, i.e., the composition result, must be a finite-state transition system. Limiting $\Omega.I$ to be finite is reasonable since we assume that Malice's computational power is polynomially bounded even though he is clever in terms of hacking the network.

The global state system obtained from the CCS system composition provides an intuitive modelling of the behaviour of a protocol under the Dolev-Yao threat model. This can be observed as follows.

5.6 Intuitive CCS Modelling of Malice's Manipulation

Assume that at a certain stage principals X and Y involve in a handshake:

$$mX.P|\bar{m}Y.Q \xrightarrow{\tau} X.P|Y.Q.$$

Notice the process definition for Ω in (2). Ω is able to observe this invisible transition. Therefore, the global state will have the following transition:

$$\Omega|(mX.P|\bar{m}Y.Q) \xrightarrow{\tau} \Omega|(X.P|Y.Q). \tag{3}$$

This transition is permitted by applying rule "Interleaving" and "Handshake" with noticing that many invisible transitions is the same as one invisible transition:

$$\xrightarrow{\tau}\xrightarrow{\tau}\cdots\xrightarrow{\tau} = \xrightarrow{\tau}.$$

By "Absorption", "Associativity" and "Commutativity" axioms, the transition in (3) is equivalent to the following

$$(\Omega|mX.P)|(\Omega|\bar{m}Y.Q) \xrightarrow{\tau} (\Omega|X.P)|(\Omega|Y.Q). \tag{4}$$

Further noticing (2) that Ω is able to receive m and is subsequently able to send $\bar{m}$, the transition in (4) can actually be considered as an interleaving of the following two transitions

$$\Omega|mX.P| \xrightarrow{\tau} \Omega|X.P \tag{5}$$

(this models Ω intercepts m from X) and

$$\Omega|\bar{m}Y.Q \xrightarrow{\tau} \Omega|Y.Q \tag{6}$$

(this models Ω injects $\bar{m}$ to Y).

Finally, we should notice that protocol messages are variables (they are not constant!). Therefore, under some conditions (to be given in a moment) the following two transitions are permeable by the global state system:

$$\Omega|mX.P| \xrightarrow{\tau} \Omega|X.P$$

and

$$\Omega|\bar{m}'Y.Q \xrightarrow{\tau} \Omega|Y.Q.$$

The conditions which permit us to use the final two transitions in places of (5) and (6) are

i) Y cannot tell the difference between m and m', and
ii) m' is available to Ω.

These two conditions are frequently met in many flawed cryptographic protocols so that Malice exploits them as a standard technique for attacking protocols. For example, they are met in the Needham-Schroeder authentication protocol [11] as have been demonstrated in the Denning-Sacco's attack [3]. There, m' is a replayed message injected by Malice to Bob who cannot discern it from m even after decryption.

6 Further Work

We have shown with appealing evidence that the *tau*-transition and the system composition in CCS-SOS provide a precise modelling of the principals' behaviour under the standard Dolev-Yao threat model.

As in the case of the usual process algebraic approaches (e.g., the approach of [5]), the intuition in the operational semantics (the rigorous spelling of subsystem behaviour) can be carefully abstracted to an internal (i.e., inside machine) representation at the denotational semantics level, where system composition can be built, and some partial-order or equivalence relations can be reasoned about, upon tool support (e.g., [2]). One desirable reasoning at the denotational semantic level is to show

$$\Omega|\ \text{System} \ \approx\ \text{System}$$

where $\approx$ is an equivalence relation defined over the algebraic terms. This equivalence states a desired security property: the participation of an adversary in a system does not change the system behaviour. In CCS and other similar process algebra approaches there exist well studied partial-order and equivalence relations which can be reasoned about at a denotational level. Our intuitive modelling of the Delov-Yao threat model using the *tau*-transition and the system composition operation of CCS-SOS invites further studies which consist of applying the well developed CCS formal methodologies and tools in the analysis of security and cryptographic protocols.

Acknowledgements. This work is conducted under the Fifth Framework IST Project "CASENET" (IST-2001-32446) funded by the European Union.

References

1. E.M. Clarke, S. Jha, and W. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols, *Proc. IFIP Working Conference on Programming Concepts*, and Methods (PROCOMET), June 1998.
2. CWB. The Edinburgh Concurrency Workbench, available at `http://www.dcs.ed.ac.uk/home/cwb/`
3. D. Denning and G. Sacco. Timestamps in key distribution protocols, *Communications of the ACM*, vol 24, no 8, pp. 533–536, August 1981.
4. D. Dolev and A. C. Yao. On the security of public key protocols, *Proceedings of the IEEE 22nd Annual Symposium on Foundations of Computer Science*, pp. 350–357, 1981.
5. M. Hennessy. *Algebraic Theory of Processes*, The MIT Press, 1988.
6. C.A.R. Hoare. Communicating sequential processes, *Communications of the ACM*, 21(8), 1978.
7. C.A.R. Hoare. *Communicating Sequential Processes*, Series in Computer Science, Prentice Hall International, 1985.
8. W. Marrero. *BRUTUS: A Model Checker for Security Protocols*, Ph.D. Thesis, CMU-CS-01-170, School of Computer Science, Carnegie Mellon University, December 2001.
9. R. Milner. *A Calculus of Communicating Systems*, Springer-Verlag, 1980.
10. R. Milner. *Communication and Concurrency*, Series in Computer Science, Prentice Hall International, 1989.
11. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers, *Communications of the ACM*, vol. 21, no 12, pp. 993–999, December 1978.

A Structured Operational Modelling of the Dolev-Yao Threat Model (Transcript of Discussion)

Wenbo Mao

Hewlett-Packard Laboratories

Pekka Nikander: I tried to do something that resembled this, some six years ago. I don't remember the details anymore, but I do remember that it turned out to be very hard to deal with the actual information content of the messages. I had to model somehow the entropy of the messages but, using this kind of formal model, it was very hard to make a distinction between the good messages, the good looking but bad messages, and the really bad messages. I originally tried to consider all possible messages, given that the length of the messages in real life is finite, not infinite. So the method I tried to use led to a model which in real life would have required infinite length messages, and that's not very realistic. So I had to consider the entropy of the messages, and then I failed to bring it into the system and I stopped work.

Virgil Gligor: So the question is, why do you think you'll not fall into the same pit?

Wenbo Mao: The first answer is that I haven't built a tool yet; so far it's in the brain, not in the systems, so it may be that I haven't found the problem. But I would like to answer the second point. In general with model checking, you have a state space explosion problem, but I think for secret protocols you do have some way to limit it; for example, you can say that a message should not be received twice. If it is, you can abandon the protocol.

Pekka Nikander: Well, it wasn't really the state space explosion problem, it was more fundamental. For example, when you receive a message and you receive the key for it later on, how do you model that? Because when you receive a message you can't tell from the message itself whether it's a good one, or bad, or what's the entropy of the message, and you have to deal with redundancy and stuff like that. It's very hard to do that in the kind of model that you are using here. As I said, I don't remember the details anymore, but I just have a feeling that that trap is lurking ahead somewhere. Maybe someone with a bit more experience with formal models can enlighten us...

Larry Paulson: My question's more the opposite. I would have thought people have done model checking of protocols so much, it's hard to see what more one could learn now, you know. I would have thought that just about all the problems have been solved.

Virgil Gligor: But that's probably because they haven't looked at details. So for example, to follow up Pekka's point, how do you know, when you get a message, and you have a key, and you decrypt the message, how do you know you've decrypted something meaningful as opposed to garbage.

Larry Paulson: No, they have looked at those details. I mean, this was six years ago, which was the Stone Age for this sort of work. You should ask the Oxford group[1] how they handled that problem. I'm not an expert on model checking but my general point remains that there are dozens of groups who are doing this stuff and they seem to be able to tackle almost any protocol that you can dream up.

Bruce Christianson: Yes, but a crucial point is the choice of the level of abstraction. These things are presented as if you were dealing with raw bit patterns.

Ernie Cohen: Actually the answer to that particular problem is that usually there's a pessimistic assumption that the adversary can't tell, and doesn't care, and that if there's a way for him to fall even accidentally into an attack, that's just as bad as his really knowing what he's doing. It's a conservative estimate, but it's hard to believe that you would defend the correctness of your protocol on the basis that the adversary can attack it but he might not know that he really found an attack.

Bruce Christianson: Well yes, it depends how long the protocol is. If your threat model allows the adversary to toss a coin a thousand times and the pattern of heads or tails turns out to be the private key, then if there are a thousand choices the adversary might make the exact correct ones.

Ernie Cohen: Right, except you usually assume keys and things like that are atomic. What the model doesn't consider are things like lengths.

Bruce Christianson: Exactly, yes.

[1] See http://web.comlab.ox.ac.uk/oucl/research/areas/security/index.html or *Modelling and Analysis of Security Protocols* by Peter Ryan, Steve Schneider, Michael Goldsmith, Gavin Lowe, and Bill Roscoe, Addison-Wesley, December 2000 - ISBN: 0201674718.

On Trust Establishment in Mobile *Ad-Hoc* Networks

Laurent Eschenauer, Virgil D. Gligor, and John Baras*

Electrical and Computer Engineering Department, University of Maryland
College Park, MD 20742, USA
{laurent, gligor, baras}@eng.umd.edu

Abstract. We present some properties of trust establishment in mobile, ad-hoc networks and illustrate how they differ from those of trust establishment in the Internet. We motivate these differences by providing an example of ad-hoc network use in battlefield scenarios, yet equally practical examples can be found in non-military environments. We argue that peer-to-peer networks are especially suitable to solve the problems of generation, distribution, and discovery of trust evidence in mobile ad-hoc networks, and illustrate the importance of evaluation metrics in trust establishment.

1 Introduction

We view the notion of "trust" among entities engaged in various protocols as a set of relations established on the basis of a body of supporting assurance (trust) evidence. In traditional networks, most trust evidence is generated via potentially lengthy assurance processes, distributed off-line, and assumed to be valid on long terms and certain at the time when trust relations derived from it are exercised. Trust relations established as a consequence of supporting trust evidence are often cached as certificates and as trust links (e.g., hierarchical or peer links) among the principals included in these relations or among their "home domains." Both certificates and trust relations are later used in authorizing client access to servers.

In contrast, few of these characteristics of trust relations and trust evidence are prevalent in *mobile ad-hoc networks (MANETs)*. Lack of a fixed networking infrastructure, high mobility of the nodes, limited-range and unreliability of wireless links are some of the characteristics of MANET environments that constrain the design of a trust establishment scheme. In particular, trust relations may have to be established using only on-line-available evidence, may be short-term and largely peer-to-peer, where the peers may not necessarily have a

* This work was supported in part by U.S. Army Research Office under Award No. DAAD19-01-1-0494, and by the U.S. Army Research Laboratory under Cooperative Agreement DAAD19-01-2-0011 for the Collaborative Technology Alliance for Communications and Networks.

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 47–66, 2004.

relevant "home domain" that can be placed into a recognizable trust hierarchy, and may be uncertain.

In this paper we argue that in MANETs a substantial body of trust evidence needs to be (1) generated, stored, and protected across network nodes, (2) routed dynamically where most needed, and (3) evaluated "on the fly" to substantiate dynamically formed trust relations. In particular, the management of trust evidence should allow alternate paths of trust relations to be formed and discovered using limited backtracking though the ad-hoc network, and should balance between the reinforcement of evidence that leads to "high-certainty" trust paths and the ability to discover alternate paths. We derive several design parameters for the generation and distribution of trust evidence in MANETs by analyzing salient characteristics of peer-to-peer file sharing protocols.

2 On Trust Establishment Differences between the Internet and MANETs

In this section, we review some of the basic notions of trust establishment and explore how these notions differ in the MANET environment from those in the Internet environmnet. We also derive a set of requirements for trust establishment in MANETs. Much of the theory underlying the presentation of basic notions can be found in Maurer [16], Kohlas and Maurer [13], others [15] [9]. We focus exclusively on some empirical properties of evidence for trust establishment that help differentiate the traditional Internet notions from those of MANETs.

2.1 Basic Notions of Trust Establishment

We view the process of trust establishment as the application of an evaluation metric to a body of trust evidence. The outcome of the trust establishment process is a trust relation. The evidence may be obtained on- or off-line and may include already established trust relations. An established trust relation constitutes evidence that can be used in other trust establishment processes, and can be composed with other relations to form more abstract or more general trust relations. The composition of trust relations usually requires the composition of evidence and of evidence evaluations.

An Example of Authentication-Trust Establishment. Consider the trust relation[1] "A accepts B's authentication of X", which is established between principals A, B, and X. This relation is established as the composition of two basic relations resulting from two separate trust-establishment processes; i.e., "certification authority B accepts X's authentication evidence," and "certification authority A accepts B's authentication of any principal registered by B". The first relation may be established by principal B's off-line evaluation of a body of trust evidence presented by principal X. For example, B may require several

[1] Although we focus on authentication, similar notions can be defined for trust establishment in the access control arera.

pieces of evidence attesting to X's identity. Specifically, B may require two pieces of authentication evidence from the following set: driver license, passport, employment identity card, documentation indicating current property ownership or credit-line activity. Once the trust relation is established, it is cached as (1) a certificate signed by B associating X's public key with X, and (2) a relation stored in B's "trust database" registering principal X with B. The domain of certification authority B becomes X's "home domain."

The second relation, namely "certification authority A accepts B's authentication of any principal registered by B," may be established by principal A's *off-line* evaluation of a body of trust evidence presented by principal B indicating that:

- certification authority B's authentication of the principals registered with it (e.g., X) is done using "acceptable" mechanisms and policies; and
- certification authority B's registration database, which includes principal X's registration, is protected using "acceptable" mechanisms and policies;
- certification authority B's server is managed using "acceptable" administrative, physical, and personnel policies;
- certification authority B does not have skills and interests that diverge from those of A.

Evidence regarding the "acceptability" of various mechanisms and policies is collected off-line, using potentially lengthy assurance procedures, such as those prescribed by the Common Criteria's assurance evaluation levels [8]. Certification authority A uses an evaluation metric to determine whether B's authentication mechanisms and policies are (at least) as good as his own, and the evidence used by the metric is *stable* and *long-term*. Evidence is stable if the authentication mechanisms and policies used by B do not change, either intentionally or accidentally, unbeknownst to A. Evidence is long-term, if it lasts at least as long as the process of gathering and evaluating assurance evidence, which can be of the order of weeks or months. After the trust relation "certification authority A accepts B's authentication of any principal registered by B" is established by A, it is cached (1) as a certificate associating B's public key with B that is signed by A, and (2) as a relation stored in A's "trust database" registering principal B with A. The domain of certification authority A becomes B's "home domain."

Transitivity of Trust Establishment. Trust relation "certification authority A accepts B's authentication of any principal registered by B" is clearly *reflexive* since A accepts its own authentication of principals it registers. However, should it be *transitive*? That is, should the trust establishment process be transitive? For example, if "A accepts B's authentication of any principal registered by B" and "B accepts Y's authentication of principal Z registered by Y," does it mean that "A accepts Y's authentication of principal Z registered by Y"? And if so, does this hold for any principals Y and Z?

Before accepting that transitivity should hold, A uses his "evaluation metric" to determine two properties of evidence. First, A determines that B's evaluation of Y's body of evidence is the same as (or stronger than) A's evaluation of B's

body of evidence (viz., example above). Second, A determines that B's trust relation with Y is (at least) as stable and long-term as his A's own with B. If these two properties of evidence hold for all Y's and Z's, then the more general trust relation "A accepts Y's authentication of any principal" should also hold. In practice, this general trust relation would hold for all Y's whose home domains are sub-domains of B's home domain. This is the case because B would control the adequacy, stability, and duration of Y's authentication mechanisms and policies, and hence could provide the evidence that would satisfy A's evaluation metric. However, evidence regarding Y's authentication mechanisms and policies may not pass A's evaluation metric, and A would not accept Y's authentication of any principal. For example, the evidence used in establishing B's trust relation with Y may be short-lived or unstable. In this case, Y could change its authentication policies, thereby invalidating evaluated evidence, unbeknownst to A and B. A would want to be protected from such events by denying transitivity regardless of whether B accepts Y's authentication of Z.

The principal characteristics of evidence used to establish transitive trust in the example given above are "uniformity" and "availability." Uniformity means that all evidence used to establish transitive trust satisfied the same, global, "metrics" of adequacy, stability, and long-term endurance. Availability means that all evidence could be evaluated either on-line or off-line at any time by a principal wishing to establish a trust relation.

Uncertainty in Trust Establishment. Transitive trust formed the basis for the definition of simple trust hierarchies, possibly interconnected by "peer" links. All early system designs supporting such hierarchies assumed either implicitly [15] or explicitly [9] that evidence for recommending trust from principal to principal was "uniform" and "available." In contrast, starting with Yahalom *et al.* [24], it was realized that, in general, trust evidence need not be uniform and hence could be uncertain. Pretty Good Privacy (PGP) [25] provides the first practical example where some "uncertainty" is allowed in authentication, although PGP does not support transitive trust. Later work by Kohlas and Maurer [13] formalizes the notion of evidence uncertainty and provides precise and fairly general principles for evaluating trust evidence.

Guaranteed Connectivity to Trust-Infrastructure Servers. To be scalable, Public Key Infrastructures (PKIs) establish trust among certification authorities rather than among individual principals. Transitive trust relations among certification authorities allows us to establish authentication trust among principals registered by different certification authorities, since it allows the traversal of certification authorities separating pairs of principals; i.e., the traversal of trust paths. Traversal of trust paths does not require that certification authorities be on-line permanently. Certification authorities store certificates in directories associated with "home domains" whenever trust relations are established, and hence directory hierarchies mirror trust hierarchies. Therefore, directory servers must be available and on-line permanently to enable trust path traversals by any principal at any time, whereas certification authority servers need be on-line only when trust relations are established and certificates are signed and stored in di-

rectories. Nevertheless, principals establishing trust relations or traversing directory hierarchies to establish, or verify the validity of, trust paths need guaranteed communication connectivity to certification authority and directory servers.

2.2 Internet *vs.* Mobile *Ad-Hoc* Networks

Ad-hoc networking refers to spontaneous formation of a network of nodes without the help of any infrastructure, usually through wireless communication channels. In *ad-hoc* networks, a basic routing infrastructure emerges through the collaboration of every node with its neighbors to forward packets towards chosen destinations. This basic infrastructure is highly dynamic not just because of node mobility (which also characterizes mobile IP networks) but also because of lack of guaranteed node connectivity (which is not necessarily a characteristic of mobile IP networks). In *ad-hoc* networks, lack of guaranteed connectivity is caused by the limited-range, potentially unreliable, wireless communication. The absence of a routing infrastructure that would assure connectivity of both fixed and mobile nodes precludes supporting a stable, long-term, trust infrastructure, such as a hierarchy of trust relations among subsets of network nodes. It also constrains the trust establishment process to short, fast, on-line-only protocols using only subsets of the established trust relations, since not all nodes that established trust relations may be reachable.

Trust Establishment without a Trust Infrastructure. In general, the Internet relies on a fixed trust infrastructure of certification-authority and directory servers for both fixed and mobile nodes (i.e., Mobile IPv6 nodes). These servers must be available on-line and reachable by principals when needed; e.g., certification authority servers, when certificates are created and signed, and directory servers permanently.

In contrast, a fixed infrastructure of certification-authority and directory servers may not always be reachable in a MANET (viz. Section 3, scenarios 2 and 3). This is because MANETs cannot assure the connectivity required to these servers; e.g., both a mobile node and the foreign-domain nodes with which it communicates can be disconnected from the directory server storing the certificates defined in that node's home domain [2]. Therefore, MANETs cannot rely exclusively on trust relations that are represented as certificates stored in directory hierarchies, since connectivity to the required servers may not be available when needed. MANETs must support *peer-to-peer relations* defined as the outcomes of any principal's evaluation of trust evidence from *any* principals in the network, and must store these trust relations in the nodes of the *ad-hoc* network.

Short-lived, Fast, and On-line-only Trust Establishment. In the Internet, trust relations are established for the long term and are stable. This is possible if security policies and assurances do not change very often and therefore do not need to be re-evaluated frequently.

[2] Note that this is not the case for mobility in the Internet. Mobile IPv6 takes care of roaming by providing a "care of" address bound to the actual mobile address. This solution is not possible for MANETs since the home of a node and its "care of" address may be physically unreachable.

In contrast, there is little long-term stability of evidence in MANETs. The security of a mobile node may depend of its location and cannot be a priori determined. For example, node capture by an adversary becomes possible and probable in some environments such as military battlefields. Trust relations involving a captured node need to be invalidated, and new trust evidence need to be collected and evaluated to maintain node connectivity in the *ad-hoc* network. Therefore, trust relations can be short-lived and the collection and evaluation of trust evidence becomes a recurrent and relatively frequent process. This process has to be fast to avoid crippling delays in the communication system; e.g., two mobile nodes may have a short time frame to communicate because of wireless range limitations, and trust establishment should not prevent these nodes from communicating securely by imposing a slow, lengthy process. To be fast, the trust establishment process may have to be executed entirely on-line since off-line collection and evaluation of evidence is impractical; e.g., visually verifying an identity document is not possible.

Trust Establishment with Incomplete Evidence. In the Internet, it is highly improbable that some trust relation remains unavailable for extended periods of time (e.g., a certificate verification on a trust path cannot performed for a day) due to connectivity failures. Network connectivity is guaranteed through redundancy of communication links, and routes and servers are replicated to guarantee availability. In general, it is fair to assume that the entire body of evidence necessary for trust establishment is available in the Internet when needed. In contrast, node connectivity is not guaranteed in MANETs and all established evidence cannot be assumed to be available for all nodes all the time. Trust establishment has to be performed with incomplete and hence uncertain trust evidence.

In summary, trust establishment in MANETs requires protocols that are:

- peer-to-peer, independent of a pre-established trust infrastructure (i.e., certification authority and directory servers);
- short, fast, and on-line; and
- flexible and support uncertain and incomplete trust evidence.

3 An Example with Three Scenarios

In this section we present an example to motivate the requirements of trust establishment presented above. The example consists of three related battlefield scenarios. For the sake of brevity, we omit relevant examples from non-military applications.

3.1 Scenario 1

In Figure 1 we illustrate a battlefield environment in which units of coalition of United States (US) and United Kingdom (UK) forces performs separate operations. To support these operations, various communication systems are involved,

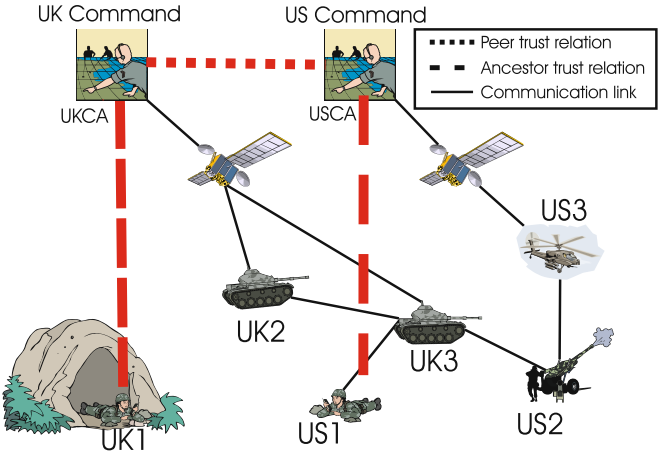


Fig. 1. A battlefield scenario. UK1 is lost and can only communicate with US1

ranging from short-range wireless (e.g., for infantry), to long-range directional wireless links (e.g., used between artillery pieces), and to satellite communication (e.g., connecting the battlefield with the US and UK operation commands). In this scenario, assume that a British unit (UK1) is lost and takes refuge in a nearby cave. UK1 needs to call for backup, but the only unit in communication range is an American unit (US1) taking part in a different operation than that of UK1. The British unit, UK1, has to authenticate itself to US1 to get access to the *ad-hoc* US network and call the UK operations command for help. UK1 requests access to the *ad-hoc* US network and presents an identity certificate signed by UKCA, the British certification authority. The US network access policy requires that any accessor presents a valid identity certificate from a US-recognized and trusted authority. Node US1 needs to decide whether the node claiming to be UK1 should be allowed access to the *ad-hoc* US network. To decide wether UK1's certificate is valid, US1 contacts the directory server at US operations command and obtains a UKCA certificate signed by USCA, the US certification authority. US1 and accepts USCA's signature on the UKCA's certificate, then accepts UKCA's signature on UK1's certificate, thereby exercising the transitive trust relations established between the US and UK operations commands and their respective units. Node US1 grants access to the *ad-hoc* US network to UK1. Note that the established trust infrastructure of the Internet helps solve UK1's problem, since all necessary trust relations (i.e., evaluated evidence) are available on-line.

3.2 Scenario 2

Assume that, due to inclement weather conditions, satellite links are unavailable. When US1 receives UK1's request and certificate signed by UKCA, it can't

contact its operations command center to retrieve UKCA's certificate from a directory server, and therefore it cannot verify the signature on UK1's certificate. However, suppose that a couple hours ago while in a different operation, a US helicopter unit, US3, visually identified the lost British unit, UK1. US3 could have *proactively* generated a certificate for UK1 and made it available in the *ad-hoc* US network. Alternately, US3 could generate and sign a certificate for UK1 now. This certificate is the only piece of evidence that could allow authentication of UK1 by US1. However, currently there is currently no scheme to specify how and when such a certificate is generated, how it can be distributed to others nodes in the network, how it should be evaluated by US1 to take its access decision and, finally, how it can be revoked by US3, if the need arise. In section 4 we present our approach on how to solve these issues.

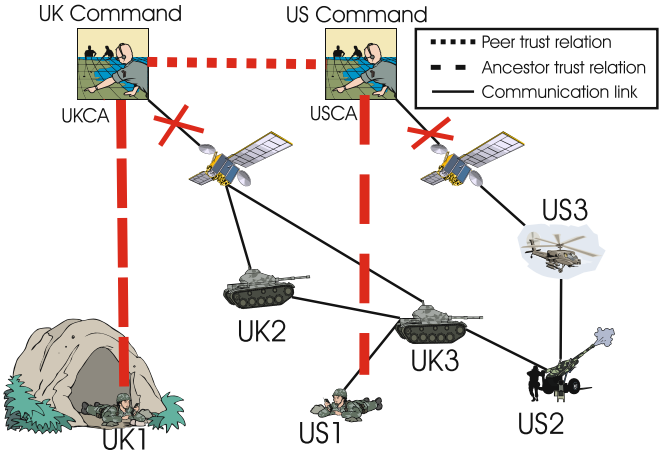


Fig. 2. A battlefield scenario. UK1 is lost and can only communicate with US1. The satellite links are down due to inclement weather

3.3 Scenario 3

Figure 3 illustrates a United Nations humanitarian convoy (UN1) that is approaching and preparing to cross a bridge separating two battlefield "zones". Before crossing the bridge to enter the new zone, UN1 must request a "zone report" from nearby military units to verify that the zone is safe. UN1 sends a request for a zone report and attaches its credentials (Table 1.b) as authentication evidence to the request. A British unit, UK3, receives the request and is in a position to issue a zone report. However, to issue the zone report, UK3 needs to apply its evaluation metric (Table 1.d and 1.e) to the presented evidence (and

the evidence already in its possession by other means) and to verify that it satisfies the policy it must enforce for providing zone reports (Table 1.a). However,

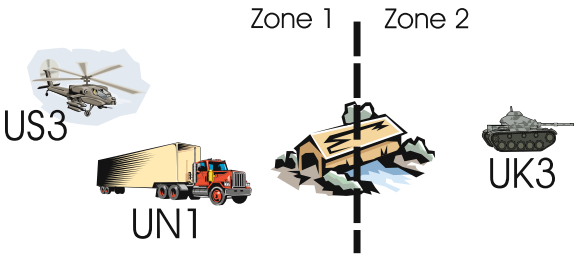


Fig. 3. A battlefield scenario

UK3 has a limited set of already established trust relations (Table 1.c) and it is not hard to see that some evidence provided by UN1 (1) is useful but cannot be verified (i.e., certificates signed by USCA and US3 cannot be verified by UK3 since it does not have a direct trust relation to USCA and US3 and the satellite links are unavailable); or (2) can be verified but is not useful (i.e., GPS1 is trusted to provide location information but the UK3 evaluation metric rates any GPS source to provide only low-confidence information whereas high-confidence information is required by the UK3 policy). Therefore, UK3 needs to collect and evaluate evidence regarding USCA and US3 using the *ad-hoc* network only, since the central directory at its operation command remains unavailable.

4 Our Approach

4.1 Generation of Trust Evidence

In our approach, any node can generate trust evidence about any other node. Evidence may be an identity, a public key, a location, an independent security assessment, or any other information required by the policy and the evaluation metric used to establish trust. Evidence is usually obtained off-line (e.g. visual identification, audio exchange [2], physical contact [20] [21], etc.), but can also be obtained on-line. When a principal generates a piece of evidence, he signs it with its own private key, specify its lifetime and makes it available to other through the network. PGP is an instance of this framework, where evidence is only a public key.

A principal may revoke a piece of evidence it produced by generating a revocation certificate for that piece of evidence and making it available to others, at any time before the evidence expires. Moreover, a principal can revoke evidence generated by others by creating contradictory evidence and distributing it. Evidence that invalidates other existant evidence can be accumulated from multiple,

Table 1. An Example of a Policy Statement, Evaluation Metric, and Credentials and Trust Relations

a. UK3's policy for providing "zone reports":
(*Role* = UK/US military ∨ UN convoy) with confidence = medium
∧(*Location*=neighbors) with confidence = high
b. UN1's request presents credentials:
Cert(*Role*=UNConvoy)$_{USCA}$
Cert(*Location*/GPS=zone2)$_{GPS1}$
Cert(*Location*/Visual=zone2)$_{US3}$
c. UK3's trust relations:
UKCA for *Role*; GPS1, UAV1, and UK1 for *Location*
d. UK3's metric for confidence evaluation of location evidence
Type(source) = GPS and source trusted → confidence = low
Type(source) = UAV and source trusted → confidence = low
Type(src1) = UAV ∧ Type(src2) = GPS
and src1 and src2 trusted → confidence = medium
Type(source) = Visual and source trusted → confidence = high
Other → confidence = null
e. UK3's metric for confidence evaluation of role evidence:
Type(source) = CA and source trusted → confidence = high
Other → confidence = null

independent, and diverses sources and will cause trust metrics to produce low confidence parameters.

It may seem dangerous to allow anyone to publish evidence within the *ad-hoc* network without control of any kind. For example, a malicious node may introduce and sign false evidence thereby casting doubt about the current trust relations of nodes and forcing them to try to verify the veracity of the (false) evidence. To protect against malicious nodes, whenever the possibility of invalidation of extant trust evidence (e.g., evidence revocation) arises, the policy must require redundant, independent pieces of (revocation) evidence from diverse sources before starting the evaluation process. Alternatively, the evaluation metric of the policy may rate the evidence provided by certain nodes as being low-confidence information. In any case, the policy and its evaluation metric can also be designed to protect against false evidence.

4.2 Distribution of Trust Evidence

Characteristics. Every principal is required to sign the pieces of evidence it produces. A principal can distribute trust evidence within the network and can even get disconnected afterwards. A producer of trust evidence does not have to be reachable at the time its evidence is being evaluated. Evidence can be replicated across various nodes to guarantee availability. This problem of evidence availability is similar to those that appear in distributed data storage systems, where

information is distributed across multiple nodes in a network, and a request for a piece of stored information is dynamically routed to the closest source.

However, trust evidence distribution is more complex than a simple "request routing" problem. A principal may need more than one answer per request, and hence *all* valid answers to a request should ideally be collected. For example, `REQUEST(Alice/location)` should return all pieces of evidence about the location of Alice. Typical distributed data storage systems do not return all valid requests; e.g. `REQUEST(my_song.mp3)` would return one file even if there are multiple versions of my_song each having different bit rates and length. Moreover a principal may simply not know what evidence to request, and hence wildcard requests have to be supported; e.g. `REQUEST(Alice/*)` should return all pieces of evidence about Alice available in the network.

Peer-to-peer file sharing for evidence distribution. The problem of evidence distribution shares many characteristics of distributed data storage systems, and yet is different. It is interesting to examine current peer-to-peer, file-sharing systems to understand their characteristics and limitations regarding trust evidence distribution. Peer-to-peer networking has received a lot of interest attention recently, particularly from the services industry [10] [17], the open-source [7], and research communities [1] [14] [22]. They evolved from very simple protocols, such as Napster (which uses a centralized index) and Gnutella (which uses request flooding) to more elaborate ones, such as Freenet (which guarantees request anonymity and uses hash-based request routing) [7] and Oceanstore (which routes requests using Plaxton trees) [14].

We analyzed Freenet as a tool for evidence distribution because of the characteristics of its request routing architecture. In particular, in Freenet requests are routed in the network instead of flooding. The routing is based on a hash of the requested keywords. Files are replicated by caching at every node. Frequently requested files are highly replicated across the network while file that are rarely requested are slowly evicted from caches. Anonymity and secrecy are guaranteed. It is not possible to know which node is requesting which file, and it is not easy to discover where a particular file is stored.

Request routing in Freenet is adaptive and improves with time; combined with the caching policy it shows an interesting locality property: information converges where needed and is forgotten where not requested. This suits particularly well the locality property of trust establishment in the MANET (a node tends to establish trust with nearby neighbors). This optimized routing allows faster distribution and revocation of pieces of evidence. However, the Freenet approach does not support wildcard requests and provides only one answer per request (due to the nature of its routing mechanism). Moreover, access to various sources of information evolves only by path reinforcement. As a consequence, some sources of information providing non-usable data are reinforced, and other sources are not discovered. The reinforcement strategy of Freenet does not preserve the diversity of information sources in the network. A new system has to be designed that shares the advantages of Freenet without exhibiting its drawbacks.

Swarm intelligence for trust evidence distribution. Swarm intelligence is a framework developed from the observation of ants' colonies. While a single ant is a very simple insect, groups of ants can cooperate and solve complex problems such as finding the shortest path to a food source or building complex structures. Ants do not communicate directly with each other; instead they induce cooperation by interacting with their environment (e.g., leaving a pheromone trail). When trying to find an optimum solution (e.g., shortest path to food source), cooperation leads to reinforcement of good solutions (positive feedback); more over, the natural decay of a pheromone trail enables regulation (negative feedback) that helps discover of new paths.

Numerous algorithms have been developed from these observations and applied to problems such as the traveling salesman, graph coloring, routing in networks [6],... Swarm intelligence is particularly suited for solving optimization problems in dynamically changing environments such as those of MANETs because of the balance between positive feedback that helps reinforce a good solution and the regulation process that enables discovery of new solutions appearing because of changes in the environment.

The problem of discovering proper sources of trust evidence in a MANET (and the problem of resource discovery in a network in general) is similar to the discovery of food supplies for an ant colony. It requires exploration of the environment with reinforcement of good solutions but also regulation that allows new sources to be discovered.

4.3 Application of an Evaluation Metric to a Body of Evidence

In specifying a trust management policy, we distinguish between a *policy decision* and a *trust metric* for practical rather than fundamental reasons. A metric is used to assign a confidence value to pieces of evidence of the same nature[3]. For instance, if we have three sources of evidence providing three different locations for Alice, how do we determine Alice's actual location and how confident are we of that determination? In contrast, a policy decision is a local procedure which, based on a set of evidence parameters and their required confidence value, outputs the outcome of the decision. In practice, policy decisions are locally enforced but may be based on trust metrics shared by other local policies. Similarly, the same policy decision may use different trust metrics (as in the case of UK3's metrics in Scenario 3 above) for different parameters. Different types of policy decisions have been proposed that apply a policy to a set of credentials and output a decision [4] [5].

Trust metrics to evaluate uncertain and incomplete sets of evidence has been an active field of research. Different "trust metrics" have been developed [16] [18] [24] and properties of these metrics have been studied [13]. However, the only practical trust metric developed and implemented has been the

[3] Different metrics may be used for different type of evidence (e.g. one may use a discrete level metric to characterize confidence in location, but a continuous metric to characterize confidence in a public key).

one of PGP [25]. Based on a very limited notion of uncertainty, this metric handles only the evaluation of trust in a chain of keys, with limited "levels of trust" (i.e. untrusted, marginal, full). There is a need to develop new trust metrics that apply to different types of evidence, not just chains of keys, are fine-grained in the sense that output wide set of uncertainty levels, and are flexible, in the sense that they can apply to incomplete sets of evidence.

5 Related Work

5.1 Pretty Good Privacy

In PGP [25], any user can sign another user's key. These signatures form a network of peer trust relations, often described as the *web of trust* [25]. The confidence in a trust path between two nodes of the web of trust is evaluated via a simple metric consisting of 4 "levels of trust" and a set of rules (e.g.: a key is marginally trusted if signed by two independent, marginally trusted, keys).

Although the PGP web of trust is fully peer-to-peer in its concepts, it is not in implementation. Public keys are published in *key servers* [19] maintaining a database of keys and discovering trust paths amongst them. This solution is efficient for the Internet but not possible for the MANET since there is no guaranteed connectivity with a key server. Hubaux *et al.* [12] propose a distributed implementation of PGP where each user stores a subset of the trust graph and proceeds to fusion of his set with other users' sets to discover trust path.

The trust metric implemented in PGP is simple and can lead to counter intuitive decision being made, as discussed by Maurer [13].

5.2 IBM's Trust Establishment System

IBM Research Laboratory developed a trust establishment framework [11] allowing the "bottom-up" emergence of a public-key infrastructure through exchange of certificates, containing various pieces of evidence about principals, and evaluation of these by a *Trust Policy Language*. When certificates about a principal are missing, they are automatically collected from peer servers. The policy language supports negative certificates, which allows complex non-monotonous policies. However, the trust policy language does not support *uncertain evidence* explicitly; as this is considered part of the policy specification.

This work is targeted to the Internet, where connectivity is guaranteed between servers. Missing certificates are collected from peer servers (either known *a priori* or referenced in other certificates). The collection mechanism is not suitable for the MANET environment were connectivity is not guaranteed. Our peer-to-peer evidence distribution mechanism would be a suitable solution to replace the certificate repositories and support the IBM's *trust engine* to provide a full peer-to-peer implementation.

5.3 The Resurrecting Duckling

Stajano and Anderson's resurrecting duckling [20] and its descendants [2] [21] represent a peer-to-peer trust establishment framework in which principals authenticate their communication channel by first exchanging keying material via an out-of-band physical contact. The goal of this approach is different from ours; i.e., it is not intended to provide peer-to-peer entity authentication, nor is it intended to handle uncertain evidence. The established trust is binary: the communication channel is either secure or is not.

6 Conclusions and Future Work

The notion of trust establishment in mobile *ad-hoc* networks (MANETs) can differ from that in the (mobile) Internet in fundamental ways. Specifically, it has the trust establishment process has to be (1) peer-to-peer, (2) short, fast, and on-line-only, and (3) flexible enough to allow uncertain and incomplete trust evidence.

We presentend a framework for trust establishment that supports the requirements for MANETs and relies on peer-to-peer file-sharing for evidence distribution through the network. The problem of evidence distribution for trust establishment is somewhat different than the usual file sharing problem in peer-to-peer networks. For this reason, we proposed to use a swarm intelligence approach for the design of trust evidence distribution instead of simply relying on an ordinary peer-to-peer, file-sharing system. In future work, we plan to evaluate the performance of swarm-based algorithms for trust evidence distribution and revocation in a MANET environment.

Finally, we also argued that the design of metrics for the evaluation of trust evidence is a crucial aspect of trust establishment in MANETs. In future work, we plan to develop a trust management scheme integrating the confidence valuation of trust evidence with real-time, policy-compliance checking.

Disclaimer

References

1. O. Babaoglu, H. Meling, and A. Montresor, "Anthill: A Framework for the Development of Agent-Based Peer-to-Peer System", Technical Report UBLCS-2001-09, University of Bologna, Italy.
2. D. Balfanz, D.K. Smetters, P. Stewart, and H. Chi Wong, "Talking To Strangers: Authentication in Ad-Hoc Wireless Networks", in Proc. of the ISOC 2002 Network and Distributed Systems Security Symposium, February 2002.

3. T. Beth, M. Borcherding, and B. Klein, "Valuation of trust in open networks", in Proc. of ESORICS 94. Brighton, UK, November 1994.
4. M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management", in Proc. of the 1996 IEEE Symposium on Security and Privacy, pages 164–173, May 1996.
5. Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis, "KeyNote: Trust management for publickey infrastructures", in Proc. Cambridge 1998 Security Protocols International Workshop, pages 59–63, 1998.
6. E. Bonabeau, M. Dorigo and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Santa Fe Institute on the Sciences of Complexity, Oxford University Press, July 1999.
7. I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," in Proc. of the International Computer Science Institute (ICSI) Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, 2000.
8. *Common Criteria for Information Technology Security Evaluation – Part 3: Security Assurance Requirements*, version 2.0, CCIB-98-028, National Institute of Standards and Technology, May 1998. *http://niap.nist.gov*
9. V. D. Gligor, S.-W. Luan, and J. N. Pato, "On inter-realm authentication in large distributed systems," in Proc. of the 1992 IEEE Symposium on Research in Security and Privacy, May 1992.
10. Gnutella, http://www.gnutellanews.com/
11. A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid, "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers," in Proc. of the 2000 IEEE Symposium on Security and Privacy, 14-17 May 2000, Berkeley, California, USA, pages 2-14
12. J.-P. Hubaux, L. Buttyan and S. Capkun, "The Quest for Security in Mobile Ad Hoc Networks," in Proc. of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001).
13. R. Kohlas and U. Maurer, "Confidence Valuation in a Public-key Infrastructure Based on Uncertain Evidence," in Proc. of Public Key Cryptography 2000, Lecture Notes in Computer Science, vol. 1751, pp. 93-112, Jan 2000.
14. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: An Architecture for Global-Scale Persistent Storage," in Proc. of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), November 2000.
15. B. W. Lampson, M. Abadi, M. Burrows, and Edward Wobber, "Authentication in distributed systems: Theory and practice,", ACM Transactions on Computer Systems, 10(4):265–310, November 1992.
16. U. Maurer, "Modelling a Public-Key Infrastructure." in Proc. ESORICS '96 (4th European Symposium on Research in Computer Security), Rome, LNCS 1146, Springer-Verlag, Berlin 1996, 325–350.
17. Napster, http://www.napster.com
18. M. K. Reiter and S. G. Stubblebine, "Toward acceptable metrics of authentication," in Proc. of the IEEE Conference on Security and Privacy, Oakland, CA, 1997.
19. M. K. Reiter and S. G. Stubblebine, "Path independence for authentication in large-scale systems," in Proc. of the 4th ACM Conference on Computer and Communications Security, April 1997.

20. F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Adhoc Wireless Networks," in Proc. of the 8th International Workshop on Security Protocols, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, 1999.
21. F. Stajano, "The resurrecting duckling – What next?," in Proc. of the 8th International Workshop on Security Protocols, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, April 2000.
22. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," in Proc. of the 2001 ACM SIGCOMM Conference, San Diego, CA, 2001, pages 149–160.
23. E. Wobber, M. Abadi, M. Burrows, and B. Lampson, "Authentication in the Taos operating system," ACM Transactions on Computer Systems, 12(1):3–32, Feb. 1994.
24. R. Yahalom, B. Klein, and T. Beth. "Trust relationships in secure systems—A distributed authentication perspective," in Proc. of the 1993 IEEE Symposium on Research in Security and Privacy, pages 150–164, May 1993.
25. P. R. Zimmermann, *The Official PGP User's Guide*, MIT Press, 1995. (http://www-mitpress.mit.edu/ mitp/recent-books/comp/pgp-user.html)
26. L. Zhou and Z. Haas, "Securing ad hoc networks," IEEE Network, 13(6):24–30, November/December 1999.

On Trust Establishment in Mobile *Ad-Hoc* Networks (Transcript of Discussion)

Laurent Eschenauer and Virgil D. Gligor

University of Maryland

Matt Blaze: This is a quick self-centered comment. The policies and credentials that you mention look like they would be quite readily implementable using KeyNote, particularly the different metrics of trust that you mention. Have you considered this?

Laurent Eschenauer: To use KeyNote, you need to collect the credentials, and that's really what we look at.

Virgil Gligor: We *are* considering implementing these things, or at least doing enough simulation to get a good grasp of what that implementation looks like. We believe that a fundamental area to work on would be this evaluation metric. At some point you have to commit to some metric. We have to look at what people have done in the past because we have to do on-line evaluation of evidence and the swarm-intelligence thing can actually help us to do this evaluation on line. We no longer need an infrastructure like the PKI, we build the infrastructure we need as we go along.

John Ioannidis: A good feature of KeyNote is that it's monotonic, so you cannot cause things not to happen by adding credentials.

Would there be any benefit in distributing all the information you receive, piggy-backed on the MANET AODV[1] routing algorithm? Are you doing that or have I missed the point?

Laurent Eschenauer: No. For the moment we are looking at maybe having an overlay network, so that the mobility does not impact me too much because the routing below me is going to take care of routing packets in the network. However, other people in the research team are looking at swarms (for example) for routing. It would be possible to combine those because both are an exploration of the network in the neighbourhood and it would diminish the overhead to do both at the same time. But for the moment we have been building the system as an overlay.

John Ioannidis: The routing also takes into consideration the appearance and disappearance of nodes.

Pekka Nikander: I was just wondering whether we have kind of chicken and egg problem here. If I understood correctly, you're trying to collect evidence for deciding whether you allow somebody to communicate with the rest of the network or not, and now you're using the same network for distributing that evidence as well.

[1] RFC 3561

Virgil Gligor: That was just one example, but it's a good example in the sense that we want it to do that. That's exactly the point.

Laurent Eschenauer: That is why, in the example we had, it is a server which is taking the load to continue requests with evidence to know if it's going to allow a request to pass in or out.

Pekka Nikander: Take a more complex situation like the US troops and the UK troops meeting at a battlefield and both of them have lost connections to the rest of the network. Both are suspicious because there should be the Rockies in between them, but they just happen to be together. How do you know that it's not suspicious, and so on?

Virgil Gligor: If you have an island and those guys were lost together and they had no access to the network and nobody sees them from the sky, then obviously they're isolated. In that case there is no sense in talking about access. I mean, that's certainly something that could happen, and life is tough. But what we are trying to do here is say that if someone somewhere generates evidence about them, then they establish this reachability.

To give you an example, if you find yourself in Singapore and you want to buy a coke with your PDA, and you are lost from the point of view of connectivity to your home domain. What do you do? When you entered the country, somebody stamped your passport - that's evidence. When you went to the hotel, they checked your passport and your credit card - that's evidence. So perhaps the hotel will authorise your PDA to buy a coke at the nearby store. That's the kind of thing.

Ross Anderson: I like the idea of using peer-to-peer networks as a basis, because the main problem with peer-to-peer networks is subversive insiders, and the main problem with battlefield equipment is battlefield capture; and these are in some sense the same thing. However, I'm very concerned at your implication that you can do quite fully distributed revocation. Suppose the rule is that someone may revoke, as suspected of capture, the equipment of someone of a lower rank. For example, a captain could revoke a lieutenant by saying "This lieutenant is now in the hands of the Iraqis, he is hereby revoked." The attack then goes as follows: a random captain is captured, the Iraqis can then send a message in respect of every lieutenant in the armed forces, saying "This lieutenant has now been captured." Thus the service denial attack is complete. The reason that you stop that normally is that there is a choke point through which the revocation has got to go such as the colonel in charge of the regiment that that particular lieutenant belongs to, and without that structure I don't see how you go forward.

Virgil Gligor: Here is the point I am trying to make. First of all, we did not address at all how revocations are authorised (the revocation policies). What we are concerned with here is simply the revocation mechanism which the kind of networks we are talking about, for example Freenet and swarm intelligence, use to direct revocation exactly where it is needed; there is no broadcasting, there are no certificate revocations that you push on anyone.

Ross Anderson: You're not answering my question, I'm saying, if random people are allowed to generate evidence that other random people are banned, there's no means of filtering that, and this becomes a mechanism for service denial attacks.

Virgil Gligor: Fake generation of evidence has to be taken into account by the evaluation metric, not by revocation. Revocation is a totally different topic. So what we address here is simply the mechanism, not whether or not a captain can revoke somebody's certificate, or anything like that. We have not addressed the policy aspect of revocation. So although your question is very relevant, it has not been addressed yet, we addressed only the mechanism for distributing revocation information fast. That is all we claim.

Laurent Eschenauer: I wanted to say something else. Your example will work if there is no single piece of other evidence about officer rank of the US, but there will be other pieces of evidence, by other people, saying that those guys are still alive, still good, still valid, and in service and doing very good work. So even if you capture one person and you use that person's credentials to generate fake evidence, it will only be evidence from one source. And that's why it's important in the metric to have independent paths of evidence and to build up the trust relationship using independent paths. That is why we need a system that distributes the most evidence to the most people so that even if you have one source of fake evidence, the rest of this good evidence is still available to everybody.

Ross Anderson: Yes, that's the problem. You are doing this in a distributed way if I, being a bad person, can call into question the honour of every soldier in the US army so that every other soldier in the US army has to start worrying about whether the guy on his left side is a traitor, and whether the guy on the right side is a spy. They would have to stop fighting for half an hour while they go and collect evidence to assure themselves that this guy's an American, and this guy's an American, and this guy's not an American, but he's a Brit so he's OK. I would have achieved a very remarkable result against your system. Now what I would suggest is that in a real life system you have to have some choke point for particular types of evidence. For example, in order to revoke a solider you must get a certificate signed by his colonel, and nobody is allowed to say bad things about the soldier except his colonel. That is how things tend to work in practice.

Laurent Eschenauer: I agree, that's in the metrics and the policy.

Virgil Gligor: There's one more thing that we didn't cover. We envisioned some of this network having some sort of a stable corner. We can harden these networks by providing a piece of the network that's almost always going to be alive and up and running. The area that we addressed with the example was the same area where we actually ran into these exceptional situations. So we haven't worked out the operational details. Clearly what you are saying is relevant, but we don't see that as being the Achilles' heel of this solution.

Bruce Christianson: What you're saying is, anybody can say anything about anybody else and you'll distribute that rapidly; whether you believe it or not is a policy decision.

Virgil Gligor: And whether we believe the evidence generated in this corner rather than evidence generated in that one.

Laurent Eschenauer: That was not the focus of our research. We really wanted to cover this evidence distribution issue.

Virgil Gligor: The trust metric is really the heart of this whole research.

Bruce Christianson: The trust metric isn't monotone. You might have said Ross is a bad guy, and I might have believed it. Then you might have said somebody else is a bad guy who I knew wasn't, so I stop believing you and I reinstate Ross.

Legally Authorized and Unauthorized Digital Evidence

Hiroshi Yoshiura[1], Kunihiko Miyazaki[1], Shinji Itoh[1], Kazuo Takaragi[1], and Ryoichi Sasaki[2]

[1] Hitachi, Ltd., Systems Development Laboratory,
292, Yoshida-cho, Totsuka-ku, Yokohama, 244-0817, Japan
{yoshiura,kunihiko,s-itoh,takara}@sdl.hitachi.co.jp
[2] Tokyo Denki University,
2-2, Kanda-Nishiki-cho, Chiyoda-ku, Tokyo 101-8457, Japan
sasaki@im.dendai.ac.jp

1 Introduction

With the electronic signature law and related laws now being put into effect, evidence-making based on digital signatures is about to come into widespread use in society. Methods for making digital evidence have not been established, however, for several important areas such as long-term maintenance of digital evidence. The authors believe that evidence can be basically classified into that which is legally authorized and that which is not, and that making a clear distinction between them will help to accelerate research and development of digital evidence-making systems.

2 Legally Authorized and Unauthorized Evidence

2.1 Legally Authorized Evidence (LAE)

LAE is evidence that is authorized by laws. A representative example in the physical world is an official patent submission. Through a patent submission, one can make LAE establishing the fact that he or she had ideas described in that patent on the submission date (note that patent submission is evidence of the existence of an idea but not evidence for its origin). Because the effectiveness of LAE is clearly defined by laws, one should always choose to make LAE from the viewpoint of certainty. People often do not do this, however, and choose instead to make unauthorized evidence because making LAE is usually quite costly. For example, a patent submission in Japan costs about a half million yens (3, 000 pounds), including the submission fee and the fees for a patent lawyer.

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 67–73, 2004.

2.2 Legally Unauthorized Evidence (LUE)

LUE is evidence that is not authorized by laws. A counterpart example of an official patent submission is making a research note. By making a research note, one can make LUE establishing the same fact that a patent submission proves. However, because the effectiveness of LUE is not defined by laws, but rather depends on the quality of the research note and the decision made by a judge, one should not choose to make LUE from the viewpoint of certainty. People often do this, however, because making LUE requires much less cost than making LAE. As an example, making a research note requires almost no cost (the cost of a notebook and about 30 minutes of time).

3 Requirements for Evidence-Making Systems

LAE and LUE pose different requirements on evidence-making systems.

(1) Systems for making LAE
These systems must be combined with laws to provide LAE. Requirements for these systems are:
   (a) The system functions (in combination with laws) must have clear legal accountability.
   (b) To achieve (a), the system functions and structure must be simple and clear even though the inside of each constituent part might be complex.
   (c) The system must be secure and its security ground must be clearly explained.
   (d) Under the condition that requirements (a), (b) and (c) are met, initial and operation costs of the system should be as low as possible.

(2) Systems for making LUE
   (a) Initial and operation costs of the system must be low because a costly LUE system is useless.
   (b) Under the condition that requirement (a) is met, the system should have clarity, accountability and security.

4 Example Guidelines for Evidence-Making Systems

4.1 Using Contextual Information

Recording an item in relation with other items is a widely used method to protect it from deleting and altering. An example of this method is to generate a digital signature from not only the target document but also from a preceding digital signature [1]. Legal researchers have the following opposing opinions on this method:

(1) Relating items introduces complexity into the system and thus causes legal disputes. For example, related items might be taken as a single item and if the validity of one component item was not proven, all other items might be taken as doubtful. Relating items thus negatively affects the maintaining of evidence and items must be recorded independently from each other.

(2) The deleting or altering of an item is easier to detect when the item is related to other items than when it is recorded alone, because any deletion or alternation would create inconsistency among the items. Relating items is thus helpful.

The authors believe that there is validity to both of these opinions. That is, opinion (a) is valid for making LAE and (b) is valid for making LUE. Because LAE (e.g., an official patent submission) is critical evidence, it must be verifiable even when it is the only item on hand. Once LAE is verifiable alone, relations with other items are not only unnecessary but even counterproductive because they introduce needless complexity. On the other hand, because the cost for making LUE must be minimal, effective measures (e.g., an authorized time stamp) cannot be applied in making it, and thus no LUE item can be very secure by itself. Relating items is therefore an effective way to strengthen LUE without entailing high costs.

4.2 Centralization and Decentralization

Centralized systems tend to be highly secure and accountable because they can be managed through rigid, uniform policies of a single organization and their operators can be strictly educated and managed. Centralized systems are thus suitable for making LAE, but the security and accountability measures taken with them tend to make their cost high.

In contrast, it is difficult to make decentralized systems highly secure and accountable because they are managed through heterogeneous policies of different organizations and it is difficult to strictly educate and mange all the operators. Due to their redundancy, however, they can be made substantially (not to say highly) secure without incurring high costs. Decentralized systems are therefore suitable for making LUE.

4.3 Long-Term Maintenance of Digital Evidence

A serious problem in this area is that certificates of public keys are valid only for a limited time (from several months to a few years) because certificate authorities cannot guarantee the security of their corresponding private keys. Thus Adigital signatures and corresponding documents cannot be verified after the validity of their corresponding public key certificates has expired. Solutions to this problem using official and private digital notaries have been proposed. This section analyzes these solutions.

(1) Solutions for LAE
Either a single official (i.e., legally authorized) digital notary or several of them may be used, and this centralization is a step in the right direction. Improving security through linking evidence items of different notaries has

been proposed but is not recommended. Security should be improved through more basic approaches such as improving underlying signature schemes and cryptosystems and security management techniques.

(2) Solutions for LUE
Private (i.e., unauthorized) digital notaries are used. Because they are used for data-center businesses, however, they are centralized rather than decentralized. Accordingly, more decentralized solutions have been recommended. Ideally, evidence should be recorded by each organization or even by each person as in the physical world. Costly measures such as authorized time stamps are not recommended. Linking evidence items in an individual archive and between different archives is recommended as a means of improving security without entailing high costs.

5 Conclusion

Legally authorized and unauthorized evidence poses different requirements on evidence-making systems. Systems for making authorized evidence must be accountable, clear, and secure rather than cost-effective. On the other hand, cost-effectiveness is the most important property in systems for making unauthorized evidence. Making a clear distinction between them therefore provides guidelines for developing evidence-making systems.

Acknowledgements. This research is supported by the Ministry of Public Management, Home Affairs, Posts and Telecommunications of Japan and the Telecommunications Advancement Organization of Japan.

References

1. T. Matsumoto, M. Iwamura, R. Sasaki, T. Matsuki: Alibi Establishment for Electronic Signature: How to Prove that You Did Not Make the Electronic Signature in Question Even When the Base Crypotsystem Was Collapsed, IPSJ-CSEC, August 2000.

Legally Authorized and Unauthorized Digital Evidence (Transcript of Discussion)

Hiroshi Yoshiura

Systems Development Laboratory, Hitachi Ltd.

Ernie Cohen: So where do existing digital time stamping systems fail? It seems like they have pretty much eliminated the expensive operations by the services they provide.

Bruce Christianson: The question is, do existing digital time stamp systems meet your criteria for an unauthorised entry?

Reply: Oh, I think so. If the existing time stamp service is not an authorised service, I think this service will eventually lose money after similar authorised services start. Or are you asking whether existing authorised systems are technically flawed?

Ernie Cohen: No, I'm saying something like Surety[1].

Reply: I don't want to specify the commercial service but I think their techniques are not bad. Of course, the technique is not public but that system is rather good, because the cost is not very high and their reliability is high. But an authorised system could also be optimised like that.

Richard Clayton: Clearly legal jurisdictions differ, and I think you've been unlucky in terms of the ones you've looked at. There are a lot of other jurisdictions with much more sensible digital signature laws. For example in Australia a digital signature is valid if a person made it, and that's basically all their law says. We have a similar effect with the law in the UK, which is that digital signature evidence cannot be excluded by the court. The court can't just look at it and say, it's a digital signature, we're not going to accept it. The implication is that if you're going to turn up with a digital signature, then you have to show them it's a valid signature. The court will look on the merits of the case as to whether or not it's a good signature. Therefore we don't have the nonsense of the law saying that my signature with this certificate has expired and is therefore invalid, whereas that signature will continue to be valid into the farness of the future. The question will solely become whether or not you believe the time stamping mechanism, i.e. that the signature was actually made during the currency of that certificate.

It is not the practice when you have certificates which expire that, come the stroke of midnight and they've expired, then you instantly publish the private keys for them. What you do is you destroy the private key so you can't make any more signatures with that particular certificate. So there is not a big problem in practice with very old signatures made under very old certificates suddenly

[1] http://www.surety.com

becoming invalid because the clock has chimed midnight on the 1st January 2005 or 2010 or 2150, or whenever it is that the certificate runs out.

Bruce Christianson: Be careful, because the crucial point is the point you made earlier about the signature: you've got to be able to prove that the signature already existed at the earlier time.

Richard Clayton: So you need a time stamping mechanism. Equally the time stamping mechanism could be that Verisign takes their private keys, holds them up to the world, and sort of hits them on the head with a hammer until they are clearly dead.

Bruce Christianson: But you pushed them back on to the time stamping mechanism.

Now the question is, what is the timestamp? Is it a digital signature by someone else?

Richard Clayton: It doesn't matter what you call it, but you have to have the mechanism because it is inherent in the digital signature idea that you must have a timestamp.

Bruce Christianson: If the time stamping mechanism is that I print out a hard copy of the signature and I go to the authority and get it stamped with a timestamp, then that's fine. But if the time stamping mechanism is that some kind of time stamping service digitally signed it, then all you've done is pushed the problem back one step.

Richard Clayton: But you concentrate what you have to demonstrate.

Bruce Christianson: I've constructed a single point of failure, yes.

Richard Clayton: But you want a single point of failure, because when eventually all the keys are obsolete you want it to destroy every single copy of that private key.

Bruce Christianson: I shouldn't have to prove that I destroyed the key. That's a very hard thing to prove, and if I could do that I wouldn't need certificates that were finite anyway.

Ernie Cohen: But digital time stamping systems are really about points of failure in that sense.

Bruce Christianson: Yes, that is true. I'm being slightly facetious but there is a real problem about using purely digital timestamps.

Richard Clayton: You can build systems with hash chains. There are all sorts of techniques you can use.

Bruce Christianson: But the question is, how do you archive old digital signatures as well as other old digital information so that the archive isn't invalidated by certificates expiring?

Ernie Cohen: Well ultimately the way of ultimate trust is that they periodically publish the hash of the whole chain in the New York Times.

Bruce Christianson & Others: Yes, yes, yes.

Pekka Nikander: I think we have also to consider quantum computing. It might be that it never becomes urgent, but if it does then we really want to have the identity in some other form.

Bruce Christianson: That's right, because all those old keys could be retrospectively broken.

Ross Anderson: I think it was three or four workshops ago we said that protocols only make sense if, after the expiration of a key, you publish the private part. If it still works then the protocol perhaps pases muster.

Bruce Christianson: If it doesn't then you have a problem, quite right.

Reply: What I want to add is that this time stamp is repeated every two years. This system will repeat the time stamp to the same certificate, because the time stamp is a kind of signature, and the certificate for the key for the time stamp will expire. So we sign repeatedly, and then we see why this signing mechanism is effective. It depends on the trust of this system: the user could re-sign but this is not acceptable, however the system re-sign is acceptable. So, at last the trust depends on the authority, and this is the same as being surety, so people trust it or not.

Ross Anderson: Here is a topical example of revocation. As we sit here the Chancellor is making his budget speech and saying how taxes will go up or down. Sometimes the tax rules change immediately so it might have been in your interest an hour ago to buy shares or, on the other hand, it might have been in your interest an hour ago to buy bonds or cigarettes. We will learn this in five minutes time. So here is an example of how to defeat such a system. You open two bank accounts and you put, say, £100,000 in each, you go to a stockbroker A, and you say, please buy me £100,000 worth of shares, and you go to the broker B and you say, please buy me £100,000 worth of bonds. You listen to the budget speech and you then cause one of the cheques to bounce as appropriate. In this way you have brought application syntax irretrievably into the act of revocation of a transaction.

Bruce Christianson: That's nice.

Shrink-Wrapped Optimism: The DODA Approach to Distributed Document Processing

Bruce Christianson and Jean F. Snook

Computer Science Department
University of Hertfordshire
Hatfield, England, Europe.

Abstract. In this paper we introduce a distributed object-based document architecture called DODA in order to illustrate a novel strategy for achieving both high availability and high integrity in the context of open processing distributed between mutually suspicious domains without a common management hierarchy.
Our approach to availability is to structure documents into small components called *folios* in such a way as to allow the maximum opportunity for concurrent processing, and to allow these components to be freely replicated and distributed. Integrity conflicts are resolved using an optimistic form of control called optimistic integrity control (OIC) applied to recoverable work units.
Our approach to security is to shrinkwrap the document components using cryptographic checksums, and to provide a set of building block components called *functionaries* which a group of users can combine in such a way as to provide each user with a means of ensuring that an agreed notion of integrity is enforced while relying upon a minimum of non-local trust.
In particular, we do not rely upon a trusted computing base or a shared system infrastructure. The local availability of document versions and of the resources to process them are completely under local user control. The lack of availability of the functionaries does not have severe consequences, and the presence of mutual suspicion makes it easier to ensure that users can trust the functionaries to provide the intended service.
A major benefit of using OIC is that it allows the integration of untrusted components such as filestores and directory servers into the system. In particular, an untrusted soft locking service can be used in order to reduce the number of concurrency conflicts, and untrusted security components can be used to screen out attempted access control violations.

Note. The text of this previously unpublished position paper is the March 1994 version.

1 Introduction

The Scylla and Charybdis of open distributed computing are lack of availability and loss of security. Availability, in the sense of transparent access being provided to data and services from anywhere in the open system and at any time,

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 74–95, 2004.

and security, in the (wide) sense of the integrity of data and services being protected from the effects of malicious or inept behaviour on the part of users or other system components, are apparently paradoxical requirements. Attempts to increase availability, for example by replicating resources such as files, tend to increase the opportunities for destroying system integrity, for example by someone updating the wrong version of a file. Conversely, conventional attempts to make a distributed system more secure are usually perceived by users as an attempt to make its resources and services less available.

Most attempts to resolve this paradox in the area of what has come to be called computer supported cooperative working (CSCW) either do not explicitly address the issue of trust, or assume global trust on the part of the participants of a large part of the computer based environment itself, including infrastructure (such as hardware, operating systems, networks) and remote services (such as file stores and name servers).

The cooperative sharing of resources (including software artifacts and data) is fundamental to the motivation for building distributed systems. But in any truly open system, autonomous management domains will never unconditionally relinquish control over their resources, since there is no "lowest common boss" in a shared organizational lattice by whom disputes over such resources could subsequently be resolved. Domain administrators will always retain a last-ditch means of reclaiming control over "their" resources.

Consequently, where computer supported cooperative working crosses organizational boundaries, it is frequently inappropriate to assume global trust on the part of participants. In practice, different members of the consortium will have different views about which other parts of the system they trust, and for what purposes. Users may entertain doubts about the honesty or competence of other consortium participants or their agents, or may be unwilling to rely upon trust to ensure the correct action of remote parts of the system belonging to other domains over which they ultimately have no administrative or management control.

On the other hand, and for the same reasons, it is usually reasonable in a genuinely open system to insist that participants should trust the infrastructure and services which they themselves choose to use and which are local to their own domain[1]. This principle of local trust is the exact reverse of that obtaining in the case of a strictly hierarchical (closed) distributed system.

Because of this locality of trust, it is unsafe to rely upon a global trusted computing base to enforce a common security policy in an open distributed system, since this would require users to trust infrastructure or services in every remote participating domain. Divergence of security domains and resource management

[1] An important (and often neglected) consequence of openness is the counterpoint to the principle of least privilege: the need to allow users themselves to confine the range of processing operations for which they are willing to accept responsibility. This allows users to reflect their degree of trust in the local facilities by indicating what uses of it they are willing to have attributed to them by users in other domains (and by implication, allows them to repudiate any other purported use.)

domains makes such an approach problematic in any case: to perform the necessary operations on the shared resources by the other policies. The hierarchy of control need not correspond to the hierarchy of trust, even if both exist.

In this paper, we attempt to explore some of these issues in the context of a distributed object-based document architecture called DODA[2]. We understand a document as, broadly, structured text with a defined operational semantics. For example, we would regard the development of a suite of software to meet a tender requiring compliance with some defined quality standard such as ISO 9000-3, by a consortium of subcontractors who are competitors with respect to other tenders, as an example of the cooperative evolution of a document.

Document processing is an interesting example to consider because documents exhibit a number of characteristics which complicate attempts to resolve the distributed system paradox by conventional approaches. Documents are typically evolved by very long term transactions (days or even weeks). The probability of transaction conflict is typically fairly low[3], but resolution of the conflicts in failed transactions frequently requires offline interaction between humans. The state of a document is easily encapsulated, and consequently documents can be freely replicated and migrated. Facilities for performing operations upon documents exist at a wide range of different locations in different domains. Consequently control of access to a document and control of the resources used to access the document cannot simply be conflated.

In this paper we are primarily concerned with mechanisms for ensuring the integrity of documents (rather than, say, their confidentiality.) We do not impose a particular notion of integrity. Central to our approach is the view that document integrity is part of the social contract between users from different domains, under which the cooperative work is taking place. Users must be free to specify and agree their own notions of integrity independently of any policy provided by shared infrastructure or services. Such a shared notion of integrity may include protection of the state of the document from deliberate tampering or accidental corruption, enforcement of access control, version control, concurrency control and/or semantic consistency, and the provision of an unforgeable audit trail. Rather than follow the usual practice of separating these issues into a hierarchy of layers, designing the layers independently and then trading off the consequences, we shall argue that it is both possible and desirable to adopt a unified approach to enforcing integrity.

We do not seek to prevent users from freely replicating and manipulating documents in their own domains using uncertified software in an arbitrary fashion. Rather, we seek to prevent a user or system service from successfully "passing off" an illegitimate or untimely version of a document to another principal as authentic.

Our approach to availability is to structure documents into small components called *folios* in such a way as to allow the maximum opportunity for con-

[2] DODA is more fully described in [10], but the terminology used there is slightly different.

[3] At least in a well-organized consortium.

current processing, and to allow these components to be freely replicated and distributed. Integrity conflicts are resolved using an optimistic form of control called optimistic integrity control (OIC) applied to recoverable work units.

Our approach to security is to shrinkwrap the document components using cryptographic checksums, and to provide a set of building block components called *functionaries* which a group of users can combine in such a way as to provide each user with a means of ensuring that an agreed notion of integrity is enforced while relying upon a minimum of non-local trust.

As we shall see, a major benefit of using OIC is that it allows the integration of untrusted components such as filestores and directory servers into the system. In particular, an untrusted soft locking service can be used in order to reduce the number of concurrency conflicts, and untrusted security components can be used to screen out attempted access control violations.

2 Document Structure

A DODA document consists of a set of components called pages, arranged in a rooted directed acyclic graph (DAG). The leaves of the DAG contain only document text. Pages which are not leaves are called *folios*. Folios are index pages which identify the contents (children) of that folio. Each page is uniquely identified by a protection number, obtained by applying a collision free hash function to the page contents[4]. The protection numbers of the children (but not their text itself) are included in the data hashed to form the protection number of the parent. Since the hash function is collision free, knowledge of the correct protection number for the base folio thus allows a user to verify the integrity of the entire document, or of any connected part of it.

Each page is an immutable object which, once created, can never be modified. A new version of a folio can be created by copying the old version, modifying the relevant fields and re-calculating the checksum. Because the hash function is collision free, the checksum will be different. Preparing a new version of a document by modifying one of its leaves thus requires making a new version of every folio which lies on a path from the modified page to the base folio, in a manner reminiscent of the Amoeba file system [7].

The protection number of the base folio (at the root of the DAG) is used to identify the current version of the document. Each folio contains the protection number of the previous version of that folio, allowing reconstruction of a linear

[4] OK it's more complicated than that. A multiplication free hash of the contents is appended to each page and the whole thing is then signed using an RSA private key to produce the protection number used to identify that page. The key ties responsibility for generation of a particular version of a folio to a specific management domain, or even to specific tamper-proof hardware within a domain. Keys are managed as described in [6]. Principals need have no access to secret keys used by their agents in remote domains, and vice versa. Of course, a domain need not include more than one principal.

version history[5]. Each document also has a unique name which remains constant throughout its history.

The folio is the unit of encapsulation, and so in order to maximize the opportunities for concurrent processing of the document, the folio structure should reflect the semantics of the document, with parents in the DAG being dependent upon their children. For example, in a document containing a piece of software, the DAG structure will resemble that of a unix "make" file, with folios containing header files being at the edge of the DAG, folios containing source for functions being parents of each included header folio, and the folio containing the object code being at the base.

Documents are object-based, so that access is via specific methods (e.g. replace, delete, insert, append) applied to specific folios, with only certain combinations of methods being allowable transactions (eg edit a source file, recompile and re-satisfy regression test.)

The methods associated with a document can also be regarded as text, since they are just code. Hence the methods can be encapsulated into folios and included in the document. This allows document processing to be generic and stateless. In particular, inclusion of variant folios containing method code allows easy accommodation of any heterogeneity within the distributed system, and method folios may be shared between documents of the same type[6]. There is no requirement to use the method code in the document to manipulate the document - users may manipulate the document by any means they choose, e.g. using a favourite editor. But the visible effect on the document must be just as if the document method had been used. This allows other principals (including functionaries associated with the validation process) to check what has been done.

An audit trail is formed by associating with each folio an incremental change record giving the protection number of the method executed to create the folio from the previous version. Each folio is signed by the principal responsible for creating the folio in the course of generating the protection number, and so the audit trail cannot be forged. Nor can principals avoid taking responsibility, since we may make it an integrity requirement that the change record exist.

A user may wish to delegate certain parts of the transaction (eg compilation, regression testing) to a service or another user, possibly in another domain, either because they trust them, or because doing this would absolve the delegating user from liability under the terms of the integrity contract. We therefore distinguish attribution of the principal responsible for carrying out a particular method (ensuring that the thing is done right) and the user responsible for authorizing the transaction (ensuring that the right thing is done.)

[5] In this paper we assume the simplest case, that of a single, universally visible, linear history for all committed versions of a particular document.

[6] In fact, we could go further and regard document types as documents in their own right, with relationships of inheritance and instantiation.

We also associate an access control list (ACL) with each folio. The ACL is also just text and so can be encapsulated and included in the document[7]. Of course, this does not by itself prevent users from preparing versions of the document which violate the access control conditions, or even from altering the access control table itself. But, since users' public key certificates are included in the ACL, any other user or functionary can (by looking at the ACL in previous, accepted versions of the document) determine whether the access control policy has been followed in the subsequent (uncommitted) history[8].

Delegation certificates (in the form of self-authenticating proxies [4]) can also be inserted in the audit trail, in order to allow users explicitly to record assumptions of trust. The propagation of delegated authority can be controlled by including propagation rights as well as access rights in the ACL.

The notion of integrity may also include discretionary elements, for example by requiring users to get their work signed off, either by other users, or by specified software checkers which act as principals in their own right. In such a case it is the presence of an appropriate certificate signed by the checker which is mandatory.

We assume in the case of each type of document that the user group (consortium) have agreed an algorithm which (given an acceptable global starting state of the document) will determine whether or not a particular set of proposed changes preserves integrity. This algorithm can itself be thought of as a particular method, called the validate method, and the code for it can be included in a folio and placed in the document with the other methods. We assume that the base folio of any document contains, in a known place, the protection number of the folio containing the validate method.

3 Document Processing

We have placed enough information in the document to allow users to satisfy themselves whether or not a particular evolution history for a document preserves integrity. It remains to describe how a group of users may identify beyond dispute which is the current version of a particular document at a particular time, and how they can ensure (in the presence of concurrent transactions by untrusted means) that the history of the currently "visible" version of the document will always satisfy the integrity criterion.

The DODA architecture does this by defining a number of *functionaries*, which are abstractions [1] trusted by certain principals to carry out particular functions. The basic philosophy in realizing functionaries is that it is infeasible to deceive a number of independent components in different mutually suspicious domains, when each component is small and self-checking.

[7] This contrasts with the server-based ACL proposed in [8].

[8] This will require a traversal of the changed parts of the document, starting from the base, but this is necessary anyway in order to check the protection numbers of altered folios.

Let us begin by imagining a document server, which is responsible for making updates to the currently visible version of a document. Given a request containing the unique name of a document, the server will return a certificate containing (i) the document name (ii) the protection number of the current version of the document (iii) a timestamp, all signed under the private key of the server.

Upon being given a new version of a document, the server will check that the proposed new version is based upon the current version, and then will extract the validation method from the base folio of the current version, and run this upon the new version. If the validation method succeeds, then the new version will be made visible as the current version to subsequent requests.

For the moment we imagine the server as a small piece of globally trusted code running on a secure hardware platform in an inaccessible place, rebooted before each service invocation, and communicating with the rest of the world by some off-line mechanism such as electronic mail, via a gateway of a type which allows all users to monitor all traffic in and out. We emphasize that this description is an abstraction of the service provided, and that any realization with the same security properties would do just as well.

A problem which is immediately apparent is that the server must not trust the validate method code to be free from side effects. Although the users for a particular document have agreed upon the integrity method for that document, it is clearly desirable for the same document server to be responsible for many different documents, of different types and with different integrity criteria (and hence different validate methods). A malicious or inept integrity method used by one group of users may in the course of execution attempt to make visible the effects of an invalid transaction upon another document of a different type owned by a different (but possibly intersecting) group of users. To prevent this from happening, the server must run the validate method in a separate (confined) address space, passing it the document folios when requested and accepting a result when the method finishes[9].

Conceptually, we regard the document server as being made up of two functionaries, a visibility server and a validation server. The validation server is given two versions of a document, one based on the other, and runs the validate method contained in the older version upon the newer, and the visibility server checks that the proposed new version is based on the current version and then updates the version table. The validation server is required to run all kinds of code, but contains no mutable state, and so can effectively be wiped clean after each method invocation. Since each method thus effectively runs in a separate address space, even on the same validation server, users of different document types do not need to certify one another's validation code. The visibility server is not stateless, but the code which it is allowed to run is very minimal, unchanging, well understood (the visibility server is effectively a name server) and is the same for all types of document, which enables a high level of certification. The advantages of this type of separation are discussed further in [5].

[9] A way of constructing an appropriate confined environment under unix without kernel changes is described in [5].

If the validation server produces a signed certificate recording the result[10], then the validation server need not be co-located with the visibility server. Because the validation server is stateless, a distributed service which provided the same abstraction in parallel, perhaps using tamper-proof boxes with different keys, would be easy enough to design, and could provide higher reliability and more localized trust. Availability *per se* is not an issue : if in doubt a user can always check a transaction by running the validate method in her own domain, although of course other principals would not be obliged to believe or accept the result.

A user attempting to make visible a new version of a document must supply with it a validation certificate. The public keys of acceptable validation servers for a document are included in the base folio of the current version of the document itself. The signature of the validation certificate must be included in the request to update the document version, and the whole request signed by the requesting user.

In fact, the problems of distributing a highly reliable name server in such a way as to require only local trust are relatively tractable [2]. Ensuring that every user trusts at least one combination of validation servers, and requiring one certificate from each such server, for example, would allow further confinement of trust by users[11].

Since transactions are long term and have a low probability of conflict, an old version of the document will probably suffice for validation checking, so the visibility server does not need to be available on-line. In a cooperative environment, there is nothing to prevent users from "unofficially" sharing changes in progress but not yet committed, in order to improve awareness of any impending conflicts[12]. Users may either bear the risk of the base version for their transaction failing to commit, or may exchange cryptographic instruments which effectively commit the version history in advance in spite of the unavailability of the visibility server. In the latter case, the integrity policy may even allow roll-back transactions to de-commit renegade transactions.

It is worth noting that all file accesses associated with the validation server can be done by untrusted software, since the integrity of each folio supplied can be checked by the validation server upon supply. Since the public key of the visibility server is assumed universally known, certificates issued by the visibility server can also be cached by untrusted software.

An attempt to commit a document transaction may fail because the transaction was validated against a version of the document which is no longer current. However, in most cases where there is no underlying conflict a simple cut and paste of the base folio index page will allow the intervening transactions to be merged. Even if this is not possible, it is usually unnecessary to reapply all or

[10] In the case of failure, the result may include a list of conflicts.

[11] The extreme case is when there is one validation server in each user's domain.

[12] Protection numbers may be exchanged by any means between parties who trust one another, for example over the telephone or in the pub.

even most of the individual methods making up the transaction, since many folios will be unchanged from the first submission.

DODA therefore includes a functionary called the submission agent whose job it is to act on behalf of a user in order to recover a failed transaction. Note that transactions may be recoverable even if they are not strictly serializable, provided the submission agent has the user's authority to vary the methods making up the transaction in the light of the new state of the document. Of course, a user may choose to act as her own submission agent (and trust no others), but this makes it more likely that subcontracting work items to principals in other domains will require the use of on-line transactions, with the associated availability and authentication problems being incurred by the untrusting user.

A user must trust her submission agent(s), even when they are not in the user's own domain. A submission agent has delegated powers from the user, so other users must trust the submission agent at least as much as they trust the delegating user[13], or must exercise control over the user's delegation rights via the ACL.

We think of the submission agent as a tamper-proof box belonging to one domain but inserted into another domain, perhaps connected via a "suspicious bridge" belonging to the second domain. Again, we emphasize that this description is an abstraction. A submission agent may be made stateless by putting the code which the user wishes it to run in the document as a method. If the user desires, the code in the document can be encrypted under the submission agent's public key to prevent other users from examining it.

Other functionaries such as archive servers or notaries [3] can also be defined, and certain aspects of their use constrained by the integrity criterion.

The separation of concerns enforced by the use of these functionaries allows DODA to reduce the effects of the open distributed system paradox to the trade-off between the availability and security of a small number of simple services. The local availability of document versions and of the resources to process them are completely under local user control. The lack of availability of functionaries in other domains does not have severe consequences, and in the presence of mutual suspicion this makes it easier to ensure that users can trust the functionaries to provide the intended service. This in turn suffices to guarantee the integrity of the cooperative work.

The server is not the authority over the document, and the DODA infrastructure does not dictate the security policy. Users retain collective control over the integrity policy, and individual control over the risks which they are willing to incur.

4 What Price Optimism?

DODA adopts a thoroughly optimistic approach to processing. Users are free to replicate, migrate, and operate upon documents as they please. The price which

[13] Although it may be easier to monitor the submission agent.

must be paid for this is that users have no guarantee that their work will not be wasted through incompatibility with work concurrently being undertaken by another principal. An objection which is often raised against the use of optimistic protocols in this respect is that they are less efficient than an appropriate fine-grained distributed locking scheme such as [12].

In fact, an optimistic protocol can usually be regarded as a semantic locking protocol in which all the locks are acquired immediately prior to committing, rather than earlier when it would have been more useful to know whether or not they were available. It would therefore seem that a locking scheme must always be more efficient.

This argument (valid though it is) does not take adequate account of the open systems paradox, in particular the consequences of lack of availability and lack of trust. The availability issue arises because we cannot assume that all domains are on-line to each other at all times[14]. Consequently there is inevitably going to be a trade-off between availability of the document page and availability of the corresponding (possibly distributed) lock. The trust issue arises because we cannot enforce the correct implementation of a lock controlling resources in a remote domain - if the application of a lock has implications for resource availability then it is only reasonable to suppose that a system administrator somewhere has the capability to remove the lock without following the protocol agreed by the consortium. We therefore need to have some code for dealing with the inevitable when it happens, and this code amounts to a form of optimistic integrity control.

Having established that we need optimistic control, we can take advantage of its presence to integrate into our system untrusted processing components, such as an uncertified distributed lock manager or protocols such as in those in [11], in order to gain performance benefits. Since we do not ultimately rely upon the lock management protocol to preserve the integrity of our document, we are free to use whatever software we please.

However, there are a number of reasons to suppose that the costs of using an optimistic approach will be low in the case of document applications : good semantic structuring of a document means a low probability of transaction conflict and therefore of validation failure, and since a transaction is represented as an unforgeable audit trail of recoverable work units, we can usually avoid re-performing most of the work units when re-performing a failed transaction. In addition, we can use other users' uncommitted versions of documents (at our own risk) as a basis for our future transactions.

This discussion on locking illustrates nicely a general principle in the design of DODA: we assume that untrusted (non-local) parts of the system will work correctly most times. Our lack of trust in something requires us to assume that it will go wrong sooner or later, and since we are not prepared to bear this risk we must have a means of recovery - but the recovery process doesn't need to be efficient if it will be infrequently invoked.

[14] A domain may periodically go offline for reasons of internal security.

An objection to shrink-wrapping which we frequently hear is that public key cryptography is too slow. We argue that this difficulty can be solved by careful structuring of documents and of the systems which process them. First, shrink-wrapping a new version of a document is an incremental process. Even though the total amount of text in a document may be very large, when a document is developed only those folios on a route from a changed page to the root of the DAG need be re-wrapped.

Since our approach is optimistic, we do not require shrinkwrapping to be done or checked immediately or on every occasion, but only when we are ready to commit a version, or (as receiver) when we are unwilling to spend more resource on processing a version with unchecked integrity. There is usually plenty of spare processing capacity and idle time on systems which the local user trusts, since our transactions are long-term[15].

It should also be remarked that a large proportion of mutable information is only relied upon for performance. Such information need not be included in the protection numbers, and hence change in this material requires no cryptographic recalculation. For example, each folio must contain a list of its children (identified by protection number) together with the number of other parents possessed by each child. The identities and locations of these other parents (which will be needed if the child is updated) do not need to be protected, in the case of identity because the hint can be checked as we go and if wrong can be repaired by exhaustive search, in the case of location because we do not need to know where the correct folios were stored in order to check their integrity. We can therefore use untrusted file-management software to increase availability.

5 Conclusion

Building open distributed systems means more than just using standardized public networks to interconnect heterogeneous hardware and software.

It is easy to enter a situation in which A's best interest is served by subverting B's artifact in such a way as to make C appear responsible. A's actions may well be actions which A (and C) are expressly permitted (trusted by B) to perform. If in fact C is responsible, but can persuade B that A has framed her, then B will trust C and not A in future. C could use this argument to blackmail A into cooperating with an even worse act of subversion, without B's awareness, unless A can be confident of proving her innocence.

[15] RSA encryption can be done in software on a 10MIP workstation (using a 500 bit modulus) at a rate of about 5Kbps, which is faster than most people can type. Integrity checking involves an RSA decrypt, which can be done in software at the rate of about 300Kbps, which is faster than most people can read. Operations (such as compiling) which require consumption or generation of text by machine can be migrated to systems with hardware cryptographic support. As was remarked in [3], a user can effectively encrypt a document in software at the decrypt rate by checking the work of an untrusted Notary with hardware support, and then signing the base folio herself, possibly using a smart card to conceal her private key.

DODA gives users a way to ensure that conformance of jointly developed documents to a mutually acceptable integrity contract cannot be irrecoverably lost, while allowing individual users to retain control over whom they trust for what purposes, and over what they themselves are willing to be held accountable for. A user can ensure that she and her fellows must each take responsibility for their own work and cannot be forced to take responsibility for anything else. Users thus have an incentive to keep their secret keys secret. A user who fails to observe proper security precautions puts only herself at risk.

DODA's end-to-end [9] integrity enforcement provides a unified approach to preventing accidental corruption, deliberate tampering, access control violations and conflicts arising from concurrency semantics. By using optimistic control, DODA provides a high locality of trust in a way which allows free replication of data and processing functions, and integration of trusted with uncertified components. The social contract is embodied in the document structure, and the means by which the contract is enforced are also held and controlled within the document. This approach is particularly suited to multi-party software development environments, when developing software conforming to quality standards, using object oriented development and certifying modules for re-use.

DODA functionaries could be realized in the form of stateless security boxes, which would then be capable of acting as property-servers, for example providing proof of whether a transaction instrument conformed to a particular policy. Such boxes (painted in pleasing pastel colours) would be of use even in a closed supposedly secure system with a traditional hierarchical lattice structure.

Acknowledgements. Our thanks to the Distributed Systems Group at Hatfield and to the Computer Security Group at the University of Cambridge Computer Laboratory for many helpful discussions. In particular, to Roger Oliver for suggesting that document processing could be interesting, and to William Harbison for insisting that trust is a fundamental issue in open distributed systems.

References

1. T.J. Gleeson, 1990, Aspects of Abstraction in Computing, PhD thesis, University of Cambridge.
2. P. Hu, 1994, Extensions to DODA, University of Hertfordshire Computer Science Technical Note
3. M.R. Low, 1992, The Notary, University of Hertfordshire Computer Science Technical Report 153
4. M.R. Low, 1993, Self-defence in Open Systems using Self-authenticating Proxies, University of Hertfordshire Computer Science Technical Report 161
5. M.R. Low and B. Christianson, 1993, Fine-grained Object Protection in Unix, ACM Operating Systems Review, 27(1)33–50
6. M.R. Low and B. Christianson, 1994, A Technique for Authentication, Access Control and Resource Management in Open Distributed Systems, Electronics Letters, 30(2)124–125

7. S.J. Mullender, 1985, Principles of Distributed Operating System Design, PhD thesis, Vrije University, Amsterdam.
8. R.G. Oliver, 1990, Protection in a Distributed Document Processing System, ACM Operating System Review, 24(2)56–66
9. J.H. Saltzer, D.P. Reed and D. Clark, 1984, End-to-End Arguments in System Design, ACM Transactions on Computer Systems **2**(4) 277–288
10. J.F. Snook, 1992, Towards Secure, Optimistic, Distributed Open Systems, University of Hertfordshire Computer Science Technical Report 151
11. R. Yahalom, 1991, Managing the Order of Transactions in Widely-distributed Data Systems, University of Cambridge Computer Laboratory Technical Report 231
12. VMS Distributed Lock Manager, VAX Cluster Manual, Digital Equipment Corporation

Shrink-Wrapped Optimism (Transcript of Talk)

Bruce Christianson

University of Hertfordshire

I'm going to talk about some work called DODA[1] that we did at Hatfield over ten years ago. In fact I talked about DODA at the very first of these protocol workshops, but some of you weren't here then, and the rest of you have probably forgotten how good that talk was. [Laughter]

The original work on DODA has been taken further at these workshops by various people over the last ten years, including Ross Anderson and Jong-Hyeon Lee with the Jikzi framework for secure publishing[2], and Geraint Price with fault-tolerant security protocols[3] – particularly the mechanisms he looked at for sharing resources. And although *PolicyMaker* was an independent piece of work, there was a certain amount of interaction between it and DODA at the fourth of these workshops.

However I'm going to talk about a rather different aspect of DODA today to what I spoke about then. My motivation this time is that people are now trying to generate programs which have verifiable integrity while using an open source model. Some are professing surprise at various properties that emerge when you try this, and I'll suggest that these properties are not particularly surprising after all.

An archetypal conflict in computer security is the tension between making data available and making data comply with whatever policy you have. This isn't an issue just with security policies. The same problem arises with concurrency control mechanisms, where a database isn't always universally globally available. So you replicate parts of it, and then whenever you do an update you're never quite sure whether or not you got the right version of the record.

It seems at first sight that you can only make one of these problems easier by making the other one harder. DODA was an attempt to eat our cake and still have it.

Software Is Text

A good example to have in your mind during this talk is some kind of software product development[4]. Imagine that you're producing some kind of software product; made up of header files, C files, configuration files, object files, executables, and so on. You've also got all the documentation for the system, which

[1] Distributed Open Document Architecture.
[2] LNCS 1796, pp 21–47.
[3] LNCS 2133, pp 155–169.
[4] You can think of DODA in this environment as providing something like a very paranoid release of *ClearCase*.

has to be consistent with the code, in accordance with your quality procedures. And you've got your regression testing suite, which consists of test harness, test scripts, correct outputs for the test runs, and so forth. Everything that you're dealing with is really text, although it's heavily structured.

You also have rules about the constraints that updates to the system must satisfy. For example you might say the development team can't update the code unless they first run the regression testing suite and get a clean run, with all the right test output.

We often talk about *availability*, by which we sometimes mean availability of the text, or we may mean availability of a service which processes the text in some kind of way. We very often talk about those two things as if they were the same. But they're not.

For example, in the product development scenario which I have just described, the developers will inevitably find a way to make their own copies of parts of the program text in which they are interested, so as to ensure the constant availability of the text to them regardless of the state of any central facility. And they will change these local copies on whatever editor they have to hand, even if it isn't one which is approved for use. So in this scenario, if the mechanism for enforcing update constraints is based upon somehow controlling all the resources used to access the text, then that mechanism is doomed.

Usually there isn't just one piece of this product, there are lots of developers working on different components which may share code. You might have some people working on what we used to call a PU-5[5], and some other people working on what we used to call a PU-2, and they're all constantly updating their code, they're shipping various releases to customers, they're getting bug reports.

When you get a bug report, you look at the bug, you think about it, you write some tests, you track the bug down. Then you book some files out, and make changes to fix the bug. But you only make the changes locally, so that you can test your fix. Effectively you build a new version of the product with some files being the locally changed ones, and others coming from the globally[6] visible product server.

Before you can release your changes, *i.e.* copy the modified files up to the product server so that they become the globally visible versions, you have to run regression tests against the new version of the product. You also write a new test to prove that the bug has gone away, and this test together with the model output is added to the regression test suite at the same time as the code changes are released. We're all familiar with this kind of thing.

That's a good example of the general case, where many different pieces of text are arranged in some kind of rooted DAG[7]. There are version control prob-

[5] Does anybody here remember SNA?

[6] Of course, 'global' is being used here as a relative term. The product server may be visible only to the product development team. Recursively, there may be a hierarchy of levels at which files are 'available', with some files at each level being possibly out-of-date or locally modified versions of files at a higher level.

[7] The Directed Acyclic Graph models the dependency relationships in a manner similar to that used by the unix `make` facility. For example, if `foo.c` includes `bar.h` then

lems: problems about compiling the correct version of the software, shipping the correct version to a customer – when the customer reports a bug you want to be trying to reproduce it on the combination that they've actually had shipped to them – and so on. Occasionally there are concurrency control problems as well: someone else who's fixing a different bug has done something that completely undermines your fix for your bug[8].

Other enemies include your own management, who are putting pressure on you to ship the product without complying with the change control procedures, and your subcontractors who actually work for a different company even though they supposedly comply with the same security policy as you. The other company is bidding against you on another contract, and would love to make your product fail provided they can do it in a way that makes it look as if it's your fault, because then the customer will buy the other product from them and not from you.

Optimization Makes Things Worse

The tension between availability and compliance is often highlighted when someone gives way to the temptation to optimise things naively. Naive performance optimisation almost always puts things inside the end-users' trust envelope without their consent, and frequently without their knowledge. For example, suppose we're getting unacceptably poor performance here in Cambridge because a particular piece of equipment in Redmond keeps falling over and needs rebooting, and we're always having to wait for it to come back on line. We might try distributing the file system and having lots of replicated files. Don't worry, there's lots of little locks[9] which you can't see that ensure that you can never get into an inconsistent state. The mistake in this example was to put the entire locking mechanism inside everybody's trust envelope.

This difficulty with a locking approach to consistency becomes particularly apparent when a policy domain crosses several different administrative domains. Security policy domains don't always map neatly onto resource management domains. And so you can bet that if there is a way of putting a lock on a resource that is in someone else's domain, and there isn't a lowest common boss, because the resource management domains are genuinely autonomous, then there's some administrator in the remote domain who has a way of taking that lock off, without the consent or even prior knowledge of the lockholder. And that means

---

foo is a parent of bar.

Each component in a DODA document includes a cryptographic checksum of each of its children in the DAG, so for instance foo will include the hash h(bar). This is the "shrink-wrapping" referred to in the title. The hash of the root thus fixes all text.

[8] Some of these are very long-term transactions, because they typically last days rather than seconds. There's lots of opportunity for nasty conflict.

[9] Does anybody here remember VMS?

that sooner or later the remote lock will break[10], so there has to be a way of recovering when that happens.

As I said before, controlling access to the document – text, data, state, or the program itself – made up of all these files, is actually a very different problem to controlling the resources used to access the files. Integrity problems like access control, concurrency control, version control, all lead to the same conclusion:

If a text is distributed widely enough that you can be reasonably sure of having availability when and where you want it, then you can't be sure that none of the pieces of hardware that can manipulate it has been subverted.

All an attacker need do is to get into one end box[11]. Since administrative domains and policy domains don't match as soon as you start delegating to a subcontractor, the attacker probably has access to a box in a vulnerable location.

Somehow we have to turn this separation, between control of a text and control of the boxes manipulating the text, into an advantage instead of a handicap.

Anybody Can Change Text

DODA doesn't even attempt to stop people from changing the text. Anybody can make any change they like, however they like. They can do whatever they want to any version of the text, even if they're not authorized, using any mechanism, even if it's not an authorized resource. And that's a *good* thing, because it enables us to solve the availability problem. What they *can't* do is to pass off an unauthorised version as an authorised version. So we want a very low level mechanism, not even for representing or manipulating security policies, but rather a mechanism for allowing people to enforce a policy, or to satisfy themselves that a particular policy had in fact been complied with.

The idea is that we have written down some sort of social contract[12], which may include the quality manual, and be supplemented by various non-aggression pacts between different teams of programmers, and with technical writers. We want a mechanism which allows us all to be sure that if anybody tries to subvert the social contract and cheat, then they will be found out.

To operationalize the social contract requires things like access control lists, and explicit representations of delegation — there's a distinction between who or what authorises something and who or what actually does it[13]. There are different *types* of things that various parties are allowed to do, and which you can think of as being like atomic transactions, except they typically take hours or days to complete. For example <release a new version of a file; recompile the product; run the regression test; check the output>. Somewhere there needs

[10] This is just another example of the basic liveness property CGMpFp, which ironically (for Computer Science) is equivalent to CGpFLp, whence CGpFLGp.

[11] Does anybody here remember Amoeba?

[12] In contrast with Rousseau, a "social contract" in DODA is a formal protocol specification.

[13] Indeed things often have to be done by someone who isn't actually permitted to authorise doing them, and vice versa. For a nice discussion of this point, see LNCS 1796, pp 48–59.

to be a representation of which *methods* comprise each transaction, what the legitimate transactions are, which principals may authorise each of those different types of transaction for each document object, and which pieces of hardware are authorised to perform each of the component methods for that transaction.

The key point is that all these things are also just text, so they can just be included in the text that is being protected. For example, when you update an ACL, that's just updating a piece of text. A method can usually be expressed as a program fragment, and that's just text[14]. An audit trail records what was actually done to the previous version in order to get the next version. That's just text too. And all these texts can be included in the document.

But a lot of mutable information doesn't need to be protected at all. For example that directed acyclic graph I showed you earlier includes information about who children's parents are, and about where particular files are stored. All that matters for correctness is whether those files have the correct checksum, not where they're stored or who their parents are.

It's very handy for performance to have that extra information, because your system works a lot more efficiently when you do. You need that ephemeral "cache" information for performance reasons, and if it's wrong then your performance will suffer. But the correctness of your system doesn't rely on the correctness of this ephemeral information. If the cached information is changed you'll discover that it's wrong, and if necessary you can recreate it by doing an exhaustive search. You *do* need to be able to work out how many parents a file has in a particular DAG – you need to know how many source files in the product include a particular header file, for example – so that you know when to stop searching. But you don't need children to include a protected list of their parents, because you could eventually find them all if you had to.

Trusted Infrastructure = Stateless Part ∪ Nameserver

The components which do need to be protected in DODA have a write-once semantics: no-one can ever update a piece of text. All they can do is create a new piece of text, and then try to persuade everyone else that the new text is now the current version. I'm not going to go deeply into the technical details of how this is done, but I'll give an outline.

Because many pieces of text are shared between DAGs corresponding to different instances (the texts of methods for example) the approach is a bit like the closed-open types the object-oriented people invented[15]. There are cryptographic checksums for each component part of the text, and the values of those checksums are included in the text of each of the parents.

Checksums for access control tables and audit trails are also included in the files to which they refer. Finally the *name* of the document object has to be linked to the checksum of the root component of the current version.

[14] You don't actually have to execute the program text in order to carry out the transaction, but the effect must be the same as if you did.

[15] Although that came later than DODA.

It is really quite startling how very little globally trusted infrastructure is needed in order to enforce a mechanism of this kind. There has to be a way of determining whether a modified version `X'` of the document is a legitimate update to a predecessor version `X`. Remember that `X'` may include in it an audit trail that says what was done to `X` to get `X'`. So imagine a routine `Valid (X,X')` which checks whether `X'` really is what you got by applying to `X` the update that's recorded in `X'`.

Now `Valid` is just another method, and so the text of `Valid` can indeed be put (via a checksum) into the document itself, along with rules for how you update that method text.

The nice thing is that this is all completely stateless, because all the state is in the text. There's consequently no difficulty with side effects if the resources that execute these methods are shared[16]. It is usually a big headache when methods for several different types of document run on the same hardware, particularly with software that implements security mechanisms. An administration domain may include some piece of hardware, some kind of resource, that is shared between two different policy domains. Even if the two different policies are compatible[17], the software that is used to enforce one policy may not be certified to run under the other policy. The DODA approach completely avoids these problems.

The separation which this statelessness gives[18] allows DODA to push almost all of the security management software out of the trust envelope. If this software is flakey then this can cause performance problems, but not integrity problems. For example, all the fine-grain locking software is no longer in the trust envelope. If it just occasionally doesn't work, there's no problem at all.

The final thing needed is a `MakeVisible` operation. This is the only stateful operation, which actually replaces the version `X` by the version `X'`. This is reasonably straightforward to implement provided `X'` contains a checksum of `X`. The update operation requires a certificate `C` from an "appropriate authority" which attests that `Valid (X,X') = true`. Provided `X` really is the current state of the document named `Doc` then `MakeVisible (Doc,C,X')` will make `X'` the current state.

What I have just described is basically a nameserver. Now it is really hard to write pieces of code that everybody will trust, but we just about understand how to do this for a nameserver. It's less than 300 lines of code. We only need one service for all types of document, the action is the same regardless of the type of document, the structure, the content, or anything else. They can all share the same nameserver.

---

[16] *c.f.* LNCS 2133, p 169.

[17] LNCS 1796, pp 6–20.

[18] There is more detail about this separation in B. Christianson, P. Hu and J.F. Snook "File Server Architecture for an Open Distributed Document System", Chapter 4 in *Communications and Multimedia Security* (ed. R. Posch), Chapman and Hall 1995 - ISBN: 0412732602.

Michael Roe: In the same way that somebody who physically owns the machine that implements a lock can cheat by taking the lock off, someone who physically owns the machine that runs the nameserver can cheat by changing the current version of a document.

Reply: Yes, that's a good point. Everybody has to trust the name *service*, but it doesn't have to be implemented as a single *server*. It doesn't have to be a single piece of hardware, it can be a replicated service with Byzantine commitment[19]. Doing the update process may actually be a very tedious business that involves going around and getting agreement from a piece of hardware in every single administrative domain — it may take 24 hours. Almost certainly you also want to have some private agreements about the order in which people are going to do things, and which are made outside of the protocol actually enforced.

The key point is that all locking is now soft. At the end of the day, if the lock is against you and you think that it's a dangling lock, and just drive through it, you can't harm the integrity of the system. You can cause a performance problem, and there are other mechanisms to deal with that, but each user can control the risk to which they're willing to expose themselves of that happening. Which brings me to my final point.

Optimism Can Reduce the Need for Trust

The usual argument against using optimistic concurrency control (OCC) is that OCC is equivalent to using fine grain semantic locking but not asking for any locks until you're just about to commit. Consequently locks will always outperform OCC, because with locking you discover conflict right at the beginning, and so don't waste time doing processing which will be thrown away.

I want to put two counterarguments to this. The first is the argument I made a little earlier: there's no reason to prevent users from doing locking for performance reasons, but with OCC they don't have to trust the lock manager anymore.

In most systems there is quite a lot of system infrastructure, which all users are required to trust and which determines the users' level of exposure to risk. But with the DODA approach, in contrast, a user can decide, for example, "I'm not doing any more work on this product until I get confirmation that that other guy's changes have committed and released successfully." Or they might say, "I'm willing to risk another day's effort, but not more." Each user can choose: the local system is in control, and DODA assumes only that users have control over, or trust in, some very local resources that are in their own administrative domain.

The second argument is that OCC needn't actually waste processor cycles. The weak form of this argument is that usually the processing that's thrown away

[19] Only the very low level code which records protocol events need be part of the replicated nameserver. The software for Byzantine agreement, like the locking software, is now explicitly outside the trust envelope. Although in practice most of the replicated servers will probably run the same code, there is no logical need for this.

is CPU cycles which, under the locking mechanism, would have been spent doing nothing while waiting for a lock. But there is a stronger form of the argument, that use of recoverable work units can often get around the serialisation problem.

Suppose A wants to commit the transaction A1, and B wants to commit the transaction B1, and these transactions cannot be done concurrently because they involve a conflict over a change to a header file, or something like that. They can't be re-ordered in such a way that one transaction doesn't break the other's locks. So either A or B has to throw their work away. But if you think about this, particularly in the context of the example of the program software system, that usually isn't true. Usually there are alternative transactions that A or B would do. If A found out that B was doing B1, A wouldn't do A1, they'd do A2. If B found out that A was doing A1, B wouldn't do B1, they'd do B2, where:

$$B1; A2 = A1; B2.$$

Now an awful lot of the little shrink-wrapped DAG nodes which are updated by this composite transaction are actually the same regardless of which way round you go. To a first approximation, the outcome is actually a merge of the pieces. It is only things near the root of the tree that need to be redone, and that's typically a small proportion of the work. Big complicated things, like running the regression test suite, tend to be shared by several different transactions anyway[20]. Starting with this post-reversibility property[21] each user can set down the rules for the modifications to their transactions they are willing to have an agent make on their behalf, and then they can delegate those powers to a *submission agent*, which you can think of as a little mechanical piece of software that A is happy for B to run as part of B's transaction. And this allows B to change A's transaction in the necessary way.

Now most of the arguments against optimistic concurrency control go away. In fact when the DODA project started, it had nothing to do with computer security, it was intended as a way of generalising concurrency control algorithms to make them apply to things other than just databases. What we discovered was a way of rehabilitating optimistic concurrency control with the bonus – when we started thinking about how malicious components could break it – that it didn't seem to require anybody to trust anybody else, or to trust any system infrastructure, very much at all.

In summary, put the social contract in the document: if you want to, put different versions of the contract in different versions of the document, and have private agreements. Anyone can verify the document, or at least can verify the bits of the document that they're allowed to see – there may be some hashes which are the hash of something secret. Global infrastructure is relied upon but

[20] Provided two source files compile to give the same object code it doesn't matter which one we test.

[21] The requirement that $\forall A1 \forall B1 \exists A2 \exists B2 : A1; B2 = B1; A2$ is actually called *right*-reversibility by the functional analysts, because they write function composition from right to left. See for example B. Christianson, *Positive Fixed Points of Lattices under Semigroups*, NZ J Math 24(1995) 23–27.

only for performance, there is no need to trust any of its functional properties. And all transfer of trust is explicit. For example, A and B both have to trust A's submission agent, but they can both see everything that it does, and verify publicly whether or not it is conforming to the protocol.

Contractual Access Control*

Babak Sadighi Firozabadi[1] and Marek Sergot[2]

[1] Swedish Institute of Computer Science (SICS)
babak@sics.se
[2] Imperial College of Science, Technology and Medicine
mjs@doc.ic.ac.uk

Abstract. In this position paper we discuss the issue of enforcing access policies in distributed environments where there is no central system designer/administrator, and consequently no guarantee that policies will be properly implemented by all components of the system. We argue that existing access control models, which are based on the concepts of permission and prohibition, need to be extended with the concept of entitlement. Entitlement to access a resource means not only that the access is permitted but also that the controller of the resource is obliged to grant the access when it is requested. An obligation to grant the access however does not guarantee that it will be granted: agents are capable of violating their obligations. In the proposed approach we discuss a *Community Regulation Server* that not only reasons about access permissions and obligations, but also updates the normative state of a community according to the contractual performance of its interacting agents.

1 Introduction

One of the problems that underlies emerging computing technologies with highly decentralised structures, such as Peer-to-Peer, Grid Computing and Ad-Hoc Networks, is how to manage access policies to disparate resources that are not under the control of a single system designer/administrator.

In these systems, a number of individuals and/or institutions interact in a collaborative environment to create a *Virtual Community* (VC), shaped and organised according to a set of rules and policies that define how its resources can be shared among its members. A VC is also sometimes called a *Virtual Organisation*, as in [5].

A VC is usually a composition of heterogenous and independently designed subsystems with no centrally controlled enforcement of the community policies. Consequently, there is no guarantee that community policies will be followed as they are prescribed: members of a VC may fail to, or choose not to, comply with the rules of the VC. If there is no way of practical (physical) enforcement of community policies then it would be useful to have a *normative control mechanism* for their *soft enforcement*. By soft enforcement we mean that even if VC

* This work is supported by the Swedish Agency for Innovation Systems (Vinnova) as part of the Policy Based Management Project.

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 96–103, 2004.

members are practically able to avoid complying with the community policies, they can still be subject to sanctioning and remedial action as the consequence of their behaviour.

1.1 Why Traditional Access Control Models Are Not Sufficient

Existing access control models are originally designed for distributed applications operating on client-server architectures. A basic assumption for these models is that there is a centrally supervised management of the entire system such that access policies will be updated and enforced as they are prescribed. For example, when a new user is introduced then its identity and its access permissions will be added to the access control lists of the provided services. Given this assumption, the policy enforcement component is trusted always to comply with the prescribed policy (unless it develops faults). The question of what to do when a resource provider deliberately fails to comply with the system's policies does not arise.

In contrast, in a system of heterogeneous and independently designed subsystems this assumption no longer holds. Consider this example: agents a_1 and a_2 are participating in an application with no central enforcement mechanism. a_1 wants to access data d that is stored remotely with agent a_2. Upon an access request from a_1, a_2 has to decide whether to grant the access to a_1 or not. There are several possible cases:

- a_1 is permitted to access the resource, but there is no obligation on a_2 to grant that access. a_2 will not violate any policy regardless of whether it grants or denies the access.
- a_1 is not only permitted to access the resource, but is also *entitled* to it. This means that a_2 has an *obligation* to grant the access whenever a_1 requests it. A typical scenario is when a_1 is the owner of d and a_2 is the storage service provider for d. Another example is where d is owned by another agent a_3 and a_3 has authorised (or rather, entitled) a_1 to access d on a_2. a_2 violates the policy if it fails to grant access to the entitled agent a_1.
- a_1 has no permission to access d, and so a_2 is forbidden to grant the access. Note that a_2 may have the practical possibility to give access to a_1 even if it is not permitted to do so.

1.2 Entitlement

In the literature on computer security, and in computer science generally, the terms 'right' and 'permission' (and 'privilege', and others) are used interchangeably. We have chosen to use the term 'entitlement' to emphasise that we have in mind a concept stronger than mere permission.

Suppose that in a VC there is a community policy that member X makes available 15GB of its disk storage for use by other members (under certain other specified terms which we ignore for the sake of the example). Suppose that X also has its own *local* policies, to the effect that members from domain D will

not be granted access to its resources (because of some previous experiences with domain D, for example), and files containing gif images will not be stored (because of the danger of storing pornographic materials, say). Suppose now that one of the members of the VC, Y, attempts to store a file on X's disks. X denies the access because the file contains gif images. Would we say that X has thereby violated (failed to comply with) the community policies operative in VC? If the answer is 'no' then the community policy is merely that Y has *permission* to store files on X's disks. If the answer is 'yes', then Y is not only permitted to store files on X's disks but is *entitled* to do so. The policy language used to express the community policies must be capable of making the distinction.

With this distinction a server might have a local policy to the effect that access to its resource will be granted to any permitted member (including entitled ones) between certain hours, but outside those hours access will be granted only to those who are entitled.

In general, a member X of some VC will be subject to (at least) two separate sets of policies: the community policies operative in VC, and the local policies defined for X. In [6] it is assumed that the community policy in the VC must always be consistent with the local policy defined for each resource. But how could this be ensured, in a system that is composed of independently designed sub-units? It is possible to imagine applications where the assumption might hold up, for instance, when all the independent resource providers either formulate their local policies to be consistent with community policies, or specify in their own local enforcement mechanisms that in case of conflict between community policies and local policies, the community policies will take precedence. But such a remarkable degree of co-operation between all the resource providers will not be so common. Rather, it is to be expected that local policies will conflict in certain circumstances with community policies, sometimes because the resource provider is looking for a 'free ride', but also because there are some detailed local considerations (such as bad previous experiences with domain D) which would lead a resource provider to choose to violate community policies from time to time. Or suppose that the community policies require that X makes available its disk storage between 6 a.m. and midnight, but X has a local policy which restricts access to the hours of 8 p.m. and 11 p.m. How could this happen? Well, leaving aside the possibility that X is out for a free ride, perhaps when X joined the VC it was expected that accesses outside those hours would be very infrequent and therefore not worth worrying about. It may also be that a resource provider belongs to more than one VC, and finds itself in circumstances where it cannot comply with the community policies of both.

Our argument is that when dealing with access control in networks of heterogeneous systems without centralised control, the usual concepts of permission and prohibition are inadequate, and must be extended with (at least) one additional concept which we are calling *entitlement*. The question is whether a request, e.g, an access request, from an agent creates an *obligation* on the controlling agent to grant this access. There are then two further subsidiary questions.

(1) What if the controlling agent ignores the request and consequently violates its obligation? (2) Under what circumstances may one agent create or pass on entitlements to another?

2 The Approach

The idea of extending access control mechanisms to deal with entitlement relations is useful for any computing environment with heterogenous and independently managed subsystems. However, we intend to investigate the idea in the setting of Grid computing [5,4] because it provides a suitable infrastructure.

Any member of a VC can be a service provider as well as a service consumer. As a service provider, a member commits itself to provide a service under certain terms and conditions to other VC members. These terms and conditions may either be specified in the community policies of the VC, or they may be negotiated by each member separately as part of their conditions for participating in an interaction with other members, or in some combination of the two.

A VC member entering into an agreement agrees not only on what it is permitted to do, but also what it is obliged to do (what its duties are to other members, and what their entitlements are). Hence, to enforce access permissions according to an agreement we need to reason not only about who is permitted or prohibited to access a service and in what conditions, but also under what conditions a service provider is obliged to give access to its service.

In [6], the authors describe a *Community Authorisation Service* (CAS) server as a trusted third party component that is responsible for managing access policies to the community's resources. CAS keeps track of the permissions that the community grants to each individual member according to the agreements obtaining between the community and its members. Based on the access state of the VC CAS issues capabilities (signed attribute certificates) that the members can provide to a resource server as credentials with their access requests. The service provider, or its resource server, will first check whether the request should be granted according to its own local policy and then check the capability issued by CAS before it grants the access.

Here, we suggest an extended CAS that we call the Community Regulation Server (CRS) which is responsible for updating the normative state of a VC as well as the soft enforcement of its policies. By normative state of a VC we mean the access permissions of the VC members, and their obligations to provide access to their services. The normative state of a VC is derived from its community policies, the agreements entered into by the VC and its members, and the management structure of the VC, including the positions and roles of the members. By soft enforcement of VC policies we mean triggering remedial actions and perhaps updating the normative state of the VC in case of violations. Moreover, the CRS will update changes in the management structure of the VC, that is to say, the delegations of management powers among the VC members as described in [2].

3 Open Issues

In order to realise the approach above there are a number of subsidiary issues that need to be addressed. Here we sketch the main requirements.

3.1 Accountability

Any regulative system requires that the entities subject to its norms (its policies and rules) can be uniquely identified in order for them to be accountable for their behaviour.

Ensuring accountability in applications where there is no identity requirement of any kind is not possible. These application types cannot be extended with a soft control mechanism and fall outside the scope of this work. Applications that require a pseudonym identity for their users will give a weak handle for accountability ensurance. The main problem with such applications is that a user can have several identities in the system at the same time. It can also disappear after a misbehaviour and reappear with another identity. One way of achieving accountability in such systems is to introduce reputation and scoring mechanisms which give the users economic incentive to retain the same identity for a longer period of time, e.g., by increasing the quality of service for those who have been lawful citizens of the virtual community for a longer period of time.

Most applications in which the users are involved in some kind of enterprise activity will require that users participate in the business with their real identities. For example, users must often identify themselves with their public key certificates which guarantees binding between their public key and their legal identities. These system can still be very dynamic in the sense that the set of users, resources, and the relations between them change frequently.

3.2 Delegation of Entitlements

We have an existing framework [2,3,1] for the delegation of authorisations and authorities via attribute certificates, together with an implementation of the reasoning required to determine, given a record of time-stamped certificates and revocations, what are the privileges (authorisations, delegation authorities) of a given entity of the system at any time. We see no difficulty in extending the existing framework to deal with delegation of entitlements as just another kind of privilege.

There is one additional factor. In human societies it is commonplace to find organisational structures in which an individual can create an obligation on another individual, possibly its superior, to exercise one of its privileges, for example to delegate some of its power. In the present context this would correspond to an entitlement of X to oblige Y to delegate one of Y's privileges. Whether this is an important or useful feature in a computational virtual organisation remains to be investigated. Our view is that it will turn out to be important.

3.3 Violation Detection and Complaint Procedure

A basic component of the infrastructure is to monitor that services are provided in accordance with the community policies and individual agreements made between members, and to detect violations (non-compliance) as they occur. The monitoring can be active, or it can be left to the members of the VC to initiate complaint proceedings against other members when agreements are unfulfilled. Here, we need to devise protocols, and associated cryptographic mechanisms, for ensuring that proper evidence is collected of both the actions and also the lack of actions of the agents.

There is a need for designing security (cryptographic) protocols to prevent false claims by agents. This is a question of guaranteeing evidence of actions (or lack of actions) on both the requester's and the service provider's side. For example, it should not be possible for a requester to claim without justification that its request was not answered properly, and it should not be possible for the service provider to claim that it has fulfilled an obligation if it has not done so.

We believe that, as a starting point, existing cryptographic protocols can be adapted for this purpose.

3.4 Sanctioning Mechanisms

It is necessary to devise effective sanctioning mechanisms in order both to encourage agents to comply with the rules of the VC and fulfill their obligations, and to provide implementable sanctions in cases where members fail, or choose not, to comply. Sanction mechanisms can be quite simple: temporary suspension of entitlements and privileges, for example, is easily implemented, as is decrease in the level of quality of service provided. More elaborate forms of sanctioning, such as the use of 'marginal accounts' or 'bonds' can also be devised. A systematic exploration of the possibilities and their effectiveness remains to be done.

References

1. Olav Bandmann, Mads Dam, and B. Sadighi Firozabadi. Constrained Delegations. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 131–140, 2002.
2. B. Sadighi Firozabadi, M. Sergot, and O. Bandmann. Using Authority Certificates to Create Management Structures, April 2001. To appear in Proceedings of the 9th International Workshop on Security Protocols, Cambridge, UK.
3. B. Sadighi Firozabadi and Marek Sergot. Revocation Schemes for Delegated Authorities. In *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks*, pages 210–213, Monterey, California, USA, June 2002. IEEE.
4. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. http://www.globus.org/research/papers/ogsa.pdf, January 2002.

5. Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid – Enabling Scalable Virtual Organisations. *International Journal of Supercomputer Applications*, 15(3), 2001.
6. L. Pearlman, V. Welch, I. Foster, and C. Kesselman. A Community Authorisation Service for Group Collaboration. In *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks*, pages 50–59, Monterey, California, USA, June 2002. IEEE.

Contractual Access Control (Transcript of Discussion)

Babak Sadighi Firozabadi

Swedish Institute of Computer Science

Bruce Christianson: Can you delegate a prohibition?

Reply: It depends on what we mean by delegation. In the way I use delegation, yes, you can do it.

John Ioannidis: You shouldn't be able to, otherwise I'll be able to deny access to anybody else out of pure spite.

Reply: No, because you would need to have the authority to delegate the prohibition.

Bruce Christianson: This is related to the question whether I can delegate to you a right without your consent. For some rights you clearly shouldn't be able to do that. If, for example, I'm a red key holder and I delegate to Mike the right to use a red key, that may mean that he is now prohibited from using a blue key or a green key on a path with anybody else, because you have a policy which says that the keys must be held by independent people.

Confidentiality Levels and Deliberate/Indeliberate Protocol Attacks

Giampaolo Bella[1,2] and Stefano Bistarelli[3]

[1] Computer Laboratory, University of Cambridge
15 JJ Thomson Avenue, Cambridge CB3 0FD, (UK)
giampaolo.bella@cl.cam.ac.uk

[2] Dipartimento di Matematica e Informatica, Università di Catania
Viale A. Doria 6, I-95125 Catania (Italy)
giamp@dmi.unict.it

[3] Istituto di Informatica e Telematica, C.N.R.
Via G. Moruzzi 1, I-56124 Pisa, Italy
stefano.bistarelli@iit.cnr.it

Abstract. A formal definition of confidentiality is developed using soft (rather than crisp) constraints. The goal is no longer considered as a mere yes/no property as in the existing literature, but gains an extra parameter, the *security level*. The higher the security level, the stronger the goal. For example, different messages may enjoy different levels of confidentiality, and the same message may enjoy different levels of confidentiality for different principals. On this basis, the notion of *indeliberate* confidentiality attack can be captured, whereby a principal learns some message not meant for him because of someone else's tampering. The analysis of Lowe's attack on the Needham-Schroeder protocol reveals a new weakness.

1 Introduction

A major goal of security protocols is *confidentiality*, confirming that a message remains undisclosed to malicious principals. However, it must be stressed that different messages require "specific degrees of protection against disclosure" [11]. For example, a user password requires higher protection than a *session key*, which is only used for a single protocol session. Intuitively, a password ought to be "more confidential" than a session key.

Confidentiality has been essentially formalised as a mere "yes or no" property thus far, so one can just claim that a key is confidential or not. The motivation for our research was studying a finer formal notion for the goal. We have developed the notion of *l-confidentiality*, where l is the *security level* signifying the strength with which the goal is met. The security level belongs to the career set of a semiring, as we adopt semiring-based soft constraint programming. Each principal assigns his own security level to each message — different levels to different messages — expressing the principal's trust on the message. This lets us formalise that different levels of a goal are granted to different principals.

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 104–119, 2004.

The framework presented here extends and supersedes an existing kernel [1, 2]. We can conduct a *preliminary analysis* to study in detail what goals the protocol achieves in ideal conditions where no principal acts maliciously. If we are interested, we can also conduct an *empirical analysis* to study what goals the protocol achieves on *a* specific network configuration arising from the protocol execution in the real world. We have observed that Lowe's attack by C against B (after A initiated with C) causes the byproduct that B unexpectedly receives a nonce he is not entitled to know. It can be concluded that B *indeliberately* mounted a confidentiality attack on the nonce. We have discovered that B can later exploit that nonce to mount an attack against C. This outline anticipates that our notation is uniform: there is not one attacker only, but *all principals who perform, either deliberately or not, some operation that is not admitted by the protocol policy are attackers.* Our use of a semiring is loosely inspired to Denning's use of a lattice to characterising secure flows of information through computer systems [7]. The idea of using *levels* to formalise access rights is in fact due to her. Denning too sees an attack whenever an object is assigned a label worse than that initially specified, rather than relying on a single attacker.

In the following, we indicate by $\{\!|m|\!\}_K$ the ciphertext obtained encrypting message m with key K, and avoid external brackets of concatenated messages. We assume the reader to be familiar with the basic concepts of cryptography. After an outline on semiring-based SCSPs (Section 2), our framework for protocol analysis is described (Section 3). The paper continues with the presentation of the crucial SCSPs for analysing security protocols (Section 4), and with the definitions concerning confidentiality (Section 5). Then, an empirical analysis of the Needham-Schroeder protocol (Section 6) highlights the possible consequences of an indeliberate confidentiality attack. Finally, some conclusions (Section 7) are given.

2 Soft Constraints

Several formalisations of the concept of *soft constraints* are currently available [18,8,10]. In the following, we refer to that by Bistarelli et al., which is based on c-semirings [4,5]. It can be shown to generalise and express a number of other approaches to soft constraints.

A soft constraint may be seen as a constraint where each instantiation of its variables has an associated value from a partially ordered set. Combining constraints will then have to take into account such additional values, and thus the formalism has also to provide suitable operations for combination ($\times$) and comparison ($+$) of tuples of values and constraints. This is why this formalisation is based on the concept of semiring, which is just a set plus two operations.

A semiring is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ such that: 1. A is a set, the *career set*, and $\mathbf{0}, \mathbf{1} \in A$; 2. $+$ is commutative, associative and $\mathbf{0}$ is its unit element; 3. $\times$ is associative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element.

A *c-semiring* is a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ such that: $+$ is idempotent, $\mathbf{1}$ as its absorbing element and $\times$ is commutative. Let us consider the relation $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$. Then it is possible to prove that (see [4]): 1. $\leq_S$

is a partial order; 2. + and × are monotone on $\leq_S$; 3. **0** is its minimum and **1** its maximum; 4. $\langle A, \leq_S \rangle$ is a complete lattice and, for all $a, b \in A$, $a + b = lub(a, b)$. Moreover, if × is idempotent, then: + distributes over ×; $\langle A, \leq_S \rangle$ is a complete distributive lattice and × its glb.

Informally, the relation $\leq_S$ gives us a way to compare (some of the) tuples of values and constraints. In fact, when we have $a \leq_S b$, we will say that b *is better than* a. Below, $\leq_S$ will be abbreviated by $\leq$.

A *constraint system* is a tuple $CS = \langle \mathcal{S}, \mathcal{D}, \mathcal{V} \rangle$ where $\mathcal{S}$ is a c-semiring, $\mathcal{D}$ is a finite set (the domain of the variables) and $\mathcal{V}$ is an ordered set of variables. Given a semiring $\mathcal{S} = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ and a constraint system $CS = \langle \mathcal{S}, \mathcal{D}, \mathcal{V} \rangle$, a *constraint* is a pair $\langle def, con \rangle$ where $con \subseteq \mathcal{V}$ and $def : \mathcal{D}^{|con|} \to A$. Therefore, a constraint specifies a set of variables (the ones in *con*), and assigns to each tuple of values of these variables an element of the semiring.

A *soft constraint satisfaction problem* (SCSP) is a pair $\langle C, con \rangle$ where $con \subseteq \mathcal{V}$ and C is a set of constraints: *con* is the set of variables of interest for the constraint set C, which however may concern also variables not in *con*. Notice that a classical constraint satisfaction problem (CSP) [14,13] is an SCSP where the chosen c-semiring is $S_{CSP} = \langle \{false, true\}, \vee, \wedge, false, true \rangle$.

Fuzzy CSPs [8,16,17] can be modelled in the SCSP framework by choosing the c-semiring: $S_{FCSP} = \langle [0, 1], max, min, 0, 1 \rangle$. Figure 1 shows the graph representation of a fuzzy CSP. Variables and constraints are represented respectively by nodes and by undirected (unary for c_1 and c_3 and binary for c_2) arcs, and semiring values are written to the right of the corresponding tuples. The variables of interest (that is the set *con*) are represented with a double circle. Here we assume that the domain D of the variables contains only elements a and b. In the following, we omit the double circle when all the variables are of interest.

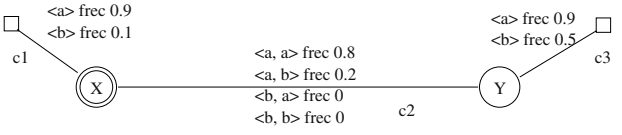


Fig. 1. A fuzzy CSP

Combining and projecting soft constraints. Given two constraints $c_1 = \langle def_1, con_1 \rangle$ and $c_2 = \langle def_2, con_2 \rangle$, their *combination* $c_1 \otimes c_2$ is the constraint $\langle def, con \rangle$ defined by $con = con_1 \cup con_2$ and $def(t) = def_1(t \downarrow_{con_1}^{con}) \times def_2(t \downarrow_{con_2}^{con})$, where $t \downarrow_Y^X$ denotes the tuple of values over the variables in Y, obtained by projecting tuple t from X to Y. In words, combining two constraints means building a new constraint involving all the variables of the original ones, and which associates to each tuple of domain values for such variables a semiring element that is obtained by multiplying the elements associated by the original constraints to the appropriate subtuples.

Given a constraint $c = \langle def, con \rangle$ and a subset I of $\mathcal{V}$, the *projection* of c over I, written $c \Downarrow_I$ is the constraint $\langle def', con' \rangle$ where $con' = con \cap I$ and

$def'(t') = \sum_{t/t\downarrow_{I\cap con}^{con}=t'} def(t)$. Informally, projecting means eliminating some variables. This is done by associating to each tuple over the remaining variables a semiring element that is the sum of the elements associated by the original constraint to all the extensions of this tuple over the eliminated variables.

In short, combination is performed via the multiplicative operation of the semiring, and projection via the additive operation.

Solutions. The *solution* of an SCSP problem $\mathsf{P} = \langle C, con\rangle$ is the constraint $Sol(\mathsf{P}) = (\bigotimes C) \Downarrow_{con}$. That is, we combine all constraints, and then project over the variables in *con*. In this way we get the constraint over *con* that is "induced" by the entire SCSP. For example, each solution of the fuzzy CSP of Figure 1 consists of a pair of domain values (that is, a domain value for each of the two variables) and an associated semiring element. Such an element is obtained by looking at the smallest value for all the subtuples (as many as the constraints) forming the pair. For example, for tuple $\langle a, a\rangle$ (that is, $x = y = a$), we have to compute the minimum between 0.9 (which is the value for $x = a$), 0.8 (which is the value for $\langle x = a, y = a\rangle$) and 0.9 (which is the value for $y = a$). Hence, the resulting value for this tuple is 0.8.

3 Constraint Programming for Protocol Analysis

This section sets out by a gentle presentation of our framework and then describes it in more detail.

Using soft constraints requires the definition of a c-semiring. Our *security semiring* (Section 3.1) is used to specify each principal's trust on the security of each message, that is each principal's *security level* on each message. The security levels range from the most secure (highest) level *unknown* to the least secure (lowest) level *public*. Intuitively, if A's security level on m is *unknown*, then no principal (included A) knows m according to A, and, if A's security level on m is *public*, then all principals potentially know m according to A. The lower A's security level on m, the higher the number of principals that A believes authorised to know m. For simplicity, we state no relation between the granularity of the security levels and the number of principals authorised to know m.

Using the security semiring, we define the *network constraint system* (Section 3.2), which represents the computer network on which the security protocols can be executed. The development of the principals' security levels from manipulation of the messages seen during the protocol sessions can be formalised as a *security entailment* (Section 3.3), that is an entailment relation between constraints. Then, given a specific protocol to analyse, we represent its assumptions in the *initial SCSP* (Section 4). All admissible network configurations arising from the protocol execution as prescribed by the protocol designers can in turn be represented in the *policy SCSP* (Section 4). We also explain how to represent any network configuration arising from the protocol execution in the real world as an *imputable SCSP* (Section 4).

Given a security level l, establishing whether our definitions of *l-confidentiality* (Section 5) holds in an SCSP requires calculating the solution

of the imputable SCSP and projecting it on certain principals of interest. The higher l, the stronger the goal.

By a *preliminary analysis*, we can study what goals the protocol achieves in ideal conditions where no principal acts maliciously. We concentrate on the policy SCSP, calculate its solution, and project it on a principal of interest. The process yields the principal's security levels, which allow us to study what goals the protocol grants to that principal in ideal conditions, and which potential attacks would be more serious than others for the principal. For example, an attack on a message whose security level is *private* is more serious than an attack on a message whose security level is *public*.

By an *empirical analysis*, we can study what goals the protocol achieves on a specific network configuration arising from the protocol execution under realistic threats. We concentrate on the corresponding imputable SCSP, calculate its solution and project it on a principal of interest: we obtain the principal's security levels on all messages. Having done the same operations on the policy SCSP, we can compare the outcomes. If some level from the imputable is lower than the corresponding level from the policy, then there is an attack in the imputable SCSP. In fact, some malicious operations contributing to the network configuration modelled by the imputable SCSP have taken place so to lower some of the security levels stated by the policy SCSP. It is important to stress that any principal might have performed, either deliberately or not, those operations.

3.1 The Security Semiring

Let n be a natural number. We define the set L of *security levels* as follows:

$$L = \{unknown,\ private,\ traded_1,\ traded_2,\ \ldots\ ,\ traded_n,\ public\}$$

We introduce an additive operator, $+_{sec}$, and a multiplicative operator, $\times_{sec}$. To allow for a compact definition of the two operators, and to simplify the following treatment, let us define a convenient double naming:
– $unknown \equiv traded_{-1}$
– $private \equiv traded_0$
– $public \equiv traded_{n+1}$

Let us consider an index i and an index j both belonging to the closed interval $[-1,\ n+1]$ of integers. We define $+_{sec}$ and $\times_{sec}$ by the following axioms.
Ax. 1: $traded_i\ +_{sec}\ traded_j\ =\ traded_{max(i,j)}$
Ax. 2: $traded_i\ \times_{sec}\ traded_j\ =\ traded_{min(i,j)}$

The structure $\mathcal{S}_{sec} = \langle L, +_{sec}, \times_{sec}, public, unknown \rangle$ can be easily verified to be a c-semiring.

3.2 The Network Constraint System

We define a constraint system $CS_n = \langle \mathcal{S}_{sec}, \mathcal{D}, \mathcal{V} \rangle$ where:
– $\mathcal{S}_{sec}$ is the security semiring (Section 3.1);
– $\mathcal{V}$ is an unbounded set of variables.

– $\mathcal{D}$ is an unbounded set of values including the empty message $\{\!|\}$ and all atomic messages, as well as all messages recursively obtained by concatenation and encryption.

We name CS_n as *network constraint system*. The elements of $\mathcal{V}$ stand for the network principals, and the elements of $\mathcal{D}$ represent all possible messages. Atomic messages typically are principal names, timestamps, nonces and cryptographic keys.

Notice that CS_n does not depend on any protocols, for it merely portrays a computer network on which any protocol can be implemented. Members of $\mathcal{V}$ will be indicated by capital letters, while members of $\mathcal{D}$ will be in small letters.

3.3 Computing the Security Levels by Entailment

Recall that each principal associates his own security levels to the messages. Those levels evolve while the principal participates in the protocol and performs off-line operations such as encryption, concatenation, decryption, and splitting.

We define four rules to compute the security levels that each principal gives to the newly generated messages. They are presented in Figure 2, where function *def* is associated to a generic constraint projected on a generic principal A. Our rules establish that the security level of a message gets somewhat lower each time the message is manipulated by encryption or decryption. Different rules could be studied if one wanted to capture other features.

Concatenation:

$$\frac{v_1, v_2 < unknown; \quad def(m_1) = v_1; \quad def(m_2) = v_2; \quad def(\{\!|m_1, m_2|\!\}) = v_3}{def(\{\!|m_1, m_2|\!\}) = (v_1 +_{sec} v_2) \times_{sec} v_3}$$

Splitting:

$$\frac{v_3 < unknown; \quad def(m_1) = v_1; \quad def(m_2) = v_2; \quad def(\{\!|m_1, m_2|\!\}) = v_3}{def(m_1) = v_1 \times_{sec} v_3; \quad def(m_2) = v_2 \times_{sec} v_3}$$

Encryption:

$$\frac{\begin{array}{l} traded_{l_1}, traded_{l_2} < unknown; \\ def(m_1) = traded_{l_1}; \ def(m_2) = traded_{l_2}; \ def(\{\!|m_1|\!\}_{m_2}) = traded_{l_3} \end{array}}{def(\{\!|m_1|\!\}_{m_2}) = (traded_{l_1+1} +_{sec} traded_{l_2}) \times_{sec} traded_{l_3}}$$

Decryption:

$$\frac{\begin{array}{l} traded_{l_2}, traded_{l_3} < unknown; \\ def(m_1) = traded_{l_1}; \ def(m_2^{-1}) = traded_{l_2}; \ def(\{\!|m_1|\!\}_{m_2}) = traded_{l_3} \end{array}}{def(m_1) = traded_{l_1} \times_{sec} traded_{l_2+1} \times_{sec} traded_{l_3}}$$

Fig. 2. Computation rules for security levels

We now define a binary relation between constraints.

Definition 1 (Relation $\vdash$). *Consider two constraints $c_1, c_2 \in C$ such that $c_1 = \langle def_1, con \rangle$ and $c_2 = \langle def_2, con \rangle$. The binary relation $\vdash$ is such that $c_1 \vdash c_2$ iff def_2 can be obtained from def_1 by a number (possibly zero) of applications of the computation rules for security levels (Figure 2).*

Theorem 1 (Relation $\vdash$ as entailment relation). *The binary relation $\vdash$ is an entailment relation.*

Proof hint. Relation $\vdash$ enjoys the reflexivity and transitivity properties, which characterise an entailment relation.

Following Theorem 1, we address the relation $\vdash$ as *security entailment*. So, if $c_1 \vdash c_2$, we say that c_1 *entails* c_2. We will use the security entailment to compute the security levels that each principal associates to the messages he derives by concatenation, splitting, encryption and decryption.

4 The Initial, Policy, and Imputable SCSPs

The designer of a protocol must develop a *policy* to accompany the protocol.

The Initial SCSP. The policy for a protocol $\mathcal{P}$ is a set of rules stating, among other things, the preconditions necessary for the protocol execution, such as which messages are public, and which messages are private for which principals. It is intuitive to capture these policy rules by our security levels (Section 3.1). Precisely, these rules can be translated into unary constraints for the network constraint system. For each principal $A \in \mathcal{V}$, we define a unary constraint that states A's security levels as follows. It associates security level *public* to those messages that are known to all, typically principal names, timestamps and public keys; level *private* to A's initial secrets, such as keys (e.g., A's long-term key if $\mathcal{P}$ uses symmetric cryptography, or A's private key if $\mathcal{P}$ uses asymmetric cryptography, or A's pin if $\mathcal{P}$ uses smart cards); level *unknown* to all remaining domain values (including, e.g., the secrets that A will invent during the protocol execution, or other principals' initial secrets).

This procedure defines what we name *initial SCSP for* $\mathcal{P}$, which specifies the principals' security levels when no session of $\mathcal{P}$ has yet started.

The Policy SCSP. The policy for a protocol $\mathcal{P}$ also specifies how the messages that must be exchanged during a session between a pair of principals are formed. The protocol designer typically writes a single step as $A \rightarrow B : m$, meaning that principal A sends message m to principal B. Each principal is typically allowed to participate in a number of protocol sessions inventing a number of secrets, namely fresh messages. Assuming both these numbers to be unbounded but finite, a finite number of *events* may take place [9]. These events consist of principals' inventing fresh messages, and principals' sending messages constructed by concatenation and/or encryption. So, when a new message is invented, the corresponding constraint is added to the store along with all constraints extracted by security entailment. No message is intercepted because no malicious principal

BUILD_POLICY_SCSP($\mathcal{P}$)

1. $\mathsf{p} \leftarrow$ *initial SCSP for* $\mathcal{P}$;
2. **for** each event ev allowed by the policy for $\mathcal{P}$ **do**
3. **if** ev = (A *invents* n, for some A and n) **then**
4. $\mathsf{p} \leftarrow \mathsf{p}$ extended with unary constraint on A that assigns *private* to n and *unknown* to all other messages;
5. **if** ev = (A *sends* m *to* B *not intercepted*, for some A, m and B) **then**
6. **let** $\langle def, con\rangle = Sol(\mathsf{p}) \Downarrow_{\{A\}} \ \wedge \ def(m) = traded_i$ **in**
7. $\mathsf{p} \leftarrow \mathsf{p}$ extended with binary constraint between A and B that assigns $traded_i$ to $\langle \{\!|\}\!|, m\rangle$ and *unknown* to all other tuples;
8. return p;

Fig. 3. Algorithm to construct the policy SCSP for $\mathcal{P}$

is assumed to be active: A's sending m to B implies that B, and B only, receives it.

We read from the protocol policy each allowed step of the form $A \rightarrow B : m$ and its informal description, which explains whether A invents m or part of it. Then, we build the *policy SCSP for* $\mathcal{P}$ by the algorithm in Figure 3. The algorithm adds new constraints to the initial SCSP according to the event that is considered. If that event is a principal A's inventing a message n, then a unary constraint is added on variable A assigning security level *private* to the domain value n (and *unknown* to all other values). If that event is a principal A's sending a message m to a principal B, then the semiring value, alias security level, associated to message m over A is considered. This level is computed by entailment (Section 3.3) whenever m is obtained by manipulation of other messages (rather than m being e.g. a fresh nonce just invented with security level *private* by the previous case of the algorithm). A binary constraint that assigns the newly computed security level to the tuple $\langle \{\!|\}\!|, m\rangle$ (and *unknown* to all other tuples) is now added to the current SCSP on the pair of variables A and B.

This reasoning is repeated for each of the unbounded (but finite) number of events allowed by the policy. When there are no more events to process, the current SCSP is returned as policy SCSP for $\mathcal{P}$, which is our formal model for the protocol.

The Imputable SCSP. A real-world network history induced by a protocol $\mathcal{P}$ must account for malicious activity by some principals. Each such history can be viewed as a sequence of events of the forms: a principal's inventing new messages, a principal's sending messages that are not intercepted, and a principal's sending messages that are intercepted. While the second event signifies that the intended recipient of a message indeed gets the message, the third signifies that some malicious principal prevents the delivery of the message that is sent.

We can model any network configuration at a certain point in any real-world network history as an SCSP by modifying the algorithm given in Figure 3

as in Figure 4 (unmodified fragments are replaced by vertical dots). The new algorithm takes as inputs a protocol $\mathcal{P}$ and a network configuration nc originated from the protocol execution. The processing of the third type of event is added: when a message is sent by A to B and is intercepted by another principal C, the corresponding constraint must be stated on the pair A, C rather than A, B.

BUILD_IMPUTABLE_SCSP($\mathcal{P}$, nc)
⋮
7.1. **if** ev = (A *sends* m *to* B *intercepted by* C, for some A, m, B and C)
then
7.2. **let** $\langle def, con\rangle = Sol(\mathsf{p}) \Downarrow_{\{A\}} \ \wedge \ def(m) = traded_i$ **in**
7.3. $\mathsf{p} \leftarrow \mathsf{p}$ extended with binary constraint between A and C that assigns $traded_i$ to $\langle \{\!\}, m\rangle$ and *unknown* to all other tuples;
⋮

Fig. 4. Algorithm to construct an imputable SCSP for $\mathcal{P}$ (fragment)

The new algorithm outputs what we name an *imputable SCSP* for $\mathcal{P}$. Clearly, there exist an unbounded number of imputable SCSPs for $\mathcal{P}$, each representing a different network configuration. Both the initial SCSP and the policy SCSP may be viewed as imputable SCSPs.

5 Formalising Confidentiality

"*Confidentiality is the protection of information from disclosure to those not intended to receive it*" [15]. This definition is often simplified into one that is easier to formalise in a model with a single attacker: a message is confidential if it is not known to the attacker. The latter definition is somewhat weaker: if a principal C who is not the attacker manages to learn a session key for A and B, the latter definition holds but the former does not. By considering all principals as potential attackers, we can capture the former definition.

We remark that a protocol may give different guarantees about its goals to different principals, so our definition of confidentiality must depend on the specific principal that is considered. Using the security levels, we develop uniform definitions of confidentiality and of confidentiality attack, which appear to capture any policy requirement. Intuitively, if a principal's security level on a message is l, then the message is *l-confidential* for the principal because the security level in fact formalises the principal's trust on the security, that is confidentiality, of the message.

Definition 2 (l-confidentiality). *Given an imputable SCSP* p *and a principal* A*, we say that there is* l-confidentiality of m for A in p *iff* $Sol(\mathsf{p}) \Downarrow_{\{A\}} (m) = l$.

Preliminary analysis on confidentiality. By studying the policy SCSP for a given protocol, we can conduct what we name a *preliminary analysis* of the protocol goals. Here, we concentrate on confidentiality.

First, we calculate the solution of the policy SCSP, and project it on some principal of interest A. Let us suppose that two messages m and m' get security levels l and l' respectively, and that $l' < l$. Thus, even if no principal acts maliciously, m' must be manipulated more than m, so A trusts that m will be less at risk than m'. We conclude that the protocol achieves a *stronger confidentiality goal on m than on m'* even if it is executed in ideal conditions. Therefore, losing m to a malicious principal would be more serious than losing m'. We address a principal's loss of m as *confidentiality attack on m.*

Empirical analysis on confidentiality. By an empirical analysis, we consider a specific real-world scenario arising from the execution of a protocol and build the corresponding imputable SCSP p. If the imputable SCSP achieves a weaker confidentiality goal of some message for some principal than the policy SCSP does, then the principal has mounted, *either deliberately or not*, a confidentiality attack on the message.

Definition 3 (Confidentiality attack).
Given a policy SCSP P*, an imputable SCSP* p *for the same protocol, and a principal A, we say that* there is a confidentiality attack by A on m in p *iff there is l-confidentiality of m in* P *for A, l'-confidentiality of m in* p *for A, and $l' < l$.*

Therefore, there is a confidentiality attack by A on m in p iff $Sol(\mathsf{P}) \Downarrow_{\{A\}} (m) < Sol(\mathsf{P}) \Downarrow_{\{A\}} (m)$. The more an attack lowers a security level, the worse that attack, so confidentiality attacks can be variously compared.

6 An Empirical Analysis of Needham-Schroeder

The protocol. Figure 5 presents the asymmetric Needham-Schroeder protocol, which is so popular that it requires little comments.

1. $A \rightarrow B : \{\!|Na, A|\!\}_{Kb}$
2. $B \rightarrow A : \{\!|Na, Nb|\!\}_{Ka}$
3. $A \rightarrow B : \{\!|Nb|\!\}_{Kb}$

Fig. 5. The asymmetric Needham-Schroeder protocol

The goal of the protocol is *authentication*: at completion of a session initiated by A with B, A should get evidence to have communicated with B and, likewise, B should get evidence to have communicated with A. Assuming that encryption is perfect and that the nonces are truly random, authentication is achieved here by confidentiality of the nonces. Indeed, upon reception of Na inside message

2, A would conclude that she is interacting with B, the only principal who could retrieve Na from message 1. In the same fashion, when B receives Nb inside message 3, he would conclude that A was at the other end of the network because Nb must have been obtained from message 2, and no-one but A could perform this operation.

Lowe discovers [12] that the protocol suffers the attack in Figure 6, whereby a malicious principal C masquerades as a principal A with a principal B, after A initiated a session with C. The attack, which sees C interleave two sessions, indicates failure of the authentication of A with B, which follows from failure of the confidentiality of Nb. The security levels of all other principals on the nonces Na and Nb are *unknown*. So, by Definition 2, those nonces are *unknown*-confidential for any principal different from A or B.

$$
\begin{aligned}
&1. \quad A \rightarrow C : \{\!|Na, A|\!\}_{Kc}\\
&1'. \quad C \rightarrow B : \{\!|Na, A|\!\}_{Kb}\\
&2'. \quad B \rightarrow A : \{\!|Na, Nb|\!\}_{Ka}\\
&2. \quad C \rightarrow A : \{\!|Na, Nb|\!\}_{Ka}\\
&3. \quad A \rightarrow C : \{\!|Nb|\!\}_{Kc}\\
&3'. \quad C \rightarrow B : \{\!|Nb|\!\}_{Kb}
\end{aligned}
$$

Fig. 6. Lowe's attack to the Needham-Schroeder Protocol

An empirical analysis. We start off by building the initial SCSP, whose fragment for principals A and B is in Figure 7 (the following only features suitable SCSP fragments pertaining to the principals of interest).

Fig. 7. Fragment of the initial SCSP for Needham-Schroeder protocol

Then, we build the policy SCSP for the protocol by Build_Policy_SCSP. Figure 8 presents the fragment pertaining to a single session between principals A and B. The figure indicates that, while A's security level on her nonce Na was initially *private*, it is now lowered to $traded_1$ by entailment because of the binary constraint formalising step 2 of the protocol. Similarly, B's security level

on Nb is now $traded_2$ though it was originally *private*. The figure omits the messages that are not relevant to the following discussion.

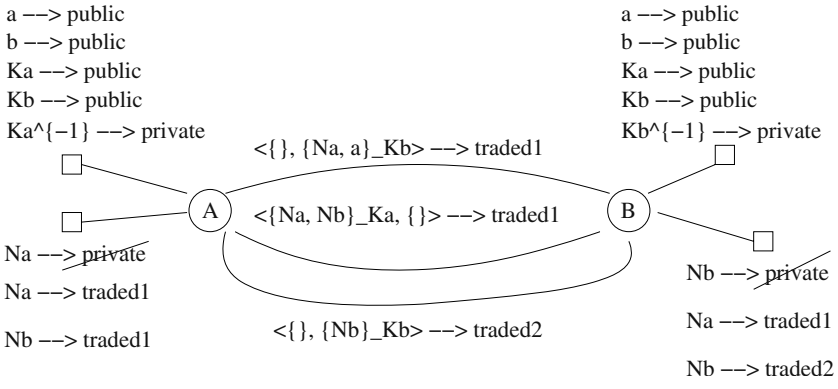


Fig. 8. Fragment of the policy SCSP for the Needham-Schroeder protocol

At this stage, we use BUILD_IMPUTABLE_SCSP to build the imputable SCSP given in Figure 9. It formalises the network configuration defined by Lowe's attack. The solution of this SCSP projected on variable C is a constraint that associates security level $traded_4$ to the nonce Nb. Following Definition 2, Nb is $traded_4$-confidential for C in this SCSP. Hence, by Definition 3, there is a deliberate confidentiality attack by C on Nb in this problem, because Nb got level *unknown* in the policy SCSP. This leads to Lowe's attack.

We discover another attack in the same problem. The problem solution projected on variable B associates security level $traded_2$ to the nonce Na, which instead got level *unknown* in the policy SCSP. This signifies that B has learnt a nonce that he was not allowed to learn by policy, that there is an indeliberate confidentiality attack by B on Na. Notice that the two attacks are uniformly formalised.

As a consequence of the former attack, Lowe reports that, if B is a bank, C can steal money from A's account as follows

$$C \rightarrow B : \{\!|Na, Nb, \text{“Transfer £1000 from } A\text{'s account to } C\text{'s”}|\!\}_{Kb}$$

and the bank B would honour the request believing it came from the account holder A. As a consequence of the attack we have discovered, if A is a bank, B can steal money from C's account as follows

$$B \rightarrow A : \{\!|Na, Nb, \text{“Transfer £1000 from } C\text{'s account to } B\text{'s”}|\!\}_{Ka}$$

and the bank A would honour the request believing it came from the account holder C. In practice, it would be sufficient that B realises what Na is for the latter crime to succeed.

There are also less serious attacks. The nonce Na is $traded_3$-confidential for A in this SCSP, while it was $traded_1$-confidential in the policy SCSP. The

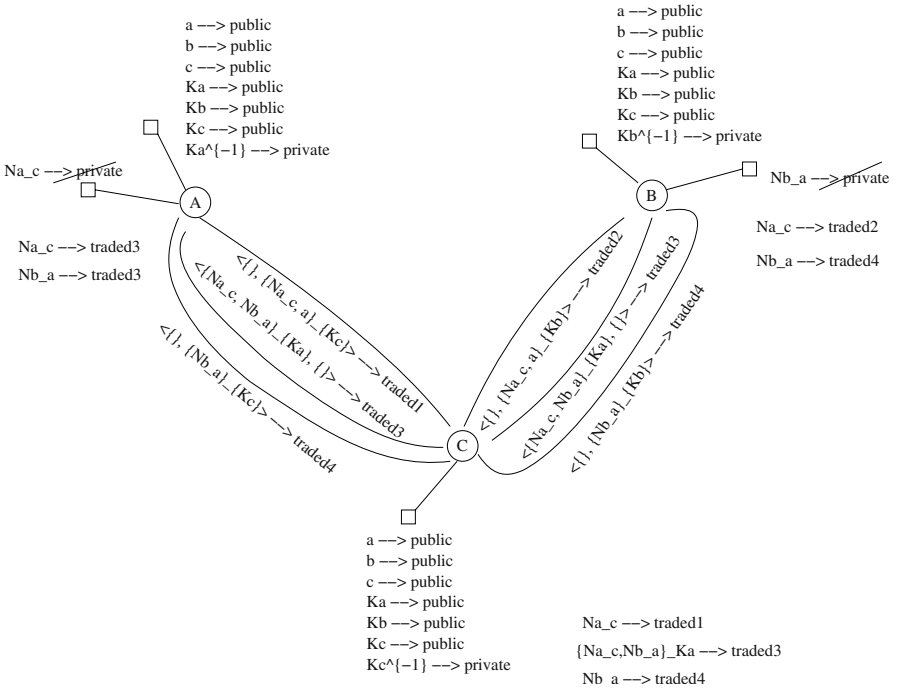


Fig. 9. Fragment of the Imputable SCSP corresponding to Lowe's attack

discrepancy highlights that the nonce has been handled differently from the policy prescription — in fact C reused it with B. Also, Nb's security level for A is $traded_3$ instead of $traded_1$ as in the policy SCSP. Similar considerations apply to Nb, whose security level for B is $traded_4$ instead of $traded_2$. This formalises C's abusive use of the nonce.

7 Conclusions

We have developed a new framework for analysing security protocols, based on a recent kernel [1,2]. Soft constraint programming allows us to conduct a fine analysis of the confidentiality that a protocol attempts to achieve. Using the security levels, we can formally claim that a configuration induced by a protocol achieves a certain level of confidentiality. That configuration may be ideal if every principal behaves according to the protocol, as formalised by the policy SCSP; or, it may arise from the protocol execution in the real world, where some principal may have acted maliciously, as formalised by an imputable SCSP. We might even compare the forms of the same goal as achieved by different protocols.

We have discovered a new attack on the asymmetric Needham-Schroeder protocol — once C masquerades as A with B, principal B indeliberately gets hold of a nonce that was not meant for him. At this stage, B might decide to exploit this extra knowledge, and begin to act maliciously. Our imputable SCSP

modelling the scenario reveals that B's security level on the nonce is lower than that allowed by the policy.

While mechanical analysis was outside our aims, we have implemented a mechanical checker for l-confidentiality on top of the existing *Constraint Handling Rule* (CHR) framework [3]. For example, when we input the policy SCSP for the Needham-Schroeder protocol and the imputable SCSP corresponding to Lowe's attack, the checker outputs

```
checking(principal(a))
checking(principal(b))
   attack(n_a, policy_level(unknown), attack_level(traded_1))
checking(principal(c))
   attack(enk(k(a),pair(n_a,n_b)), policy_level(unknown),
                                   attack_level(traded_1))
   attack(n_b, policy_level(unknown), attack_level(traded1))
```

The syntax seems to be self-explanatory. Line two reveals the new attack we have found on B, who has lowered his security level on Na from *unknown* to $traded_1$. Likewise, line three denounces that not only has C got hold of the nonce Nb but also of the message $\{\!|Na, Nb|\!\}_{Ka}$ (which was meant for A and not for B) that contains it.

At this stage, integrating our framework with model-checking tools appears to be a straightforward exercise. The entailment relation must be extended by a rule per each of the protocol messages in order to compute their security levels. Hence, our constraints would be upgraded much the way multisets are rewritten in the work by Cervesato et al. [6] (though they only focus on a single attacker and their properties are classical yes/no properties). Then, once suitable size limits are stated, the imputable SCSPs could be exhaustively generated and checked against our definitions of confidentiality.

Acknowledgements. We are indebted to Martin Abadi for invaluable suggestions, and to Michael Marte and Thom Fruewirth for the preliminary implementation of the checker. Special thanks also to Larry Paulson and to Peter Ryan for useful discussions. Research was funded by the EPSRC grant GR/R01156/01 *Verifying Electronic Commerce Protocols*, by MIUR project *Tecniche e strumenti software per l'analisi della sicurezza delle comunicazioni in applicazioni telematiche di interesse economico e sociale*, by CNR project *Strumenti, ambienti ed applicazioni innovative per la società dell'informazione* and by CSP with the project *SeTAPS*.

References

1. G. Bella and S. Bistarelli. Protocol Analysis using Soft Constraints. Invited talk at S.R.I. Security group, Menlo Park, USA, February 2001.
2. G. Bella and S. Bistarelli. Soft Constraints for Security Protocol Analysis: Confidentiality. In I. V. Ramakrishnan, editor, *Proc. of the 3rd International Symposium on Practical Aspects of Declarative Languages (PADL'01)*, volume 1990 of *LNCS*, pages 108–122. Springer-Verlag, 2001.

3. S. Bistarelli, T. Fruewirth, M. Marte, and F. Rossi. Soft Constraint Propagation and Solving in CHR. In *Proc. ACM Symposium on Applied Computing (SAC), Madrid, Spain*. ACM, mar 2002.
4. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *Journal of the ACM*, 44(2):201–236, Mar 1997.
5. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Logic Programming: Syntax and Semantics. *ACM Transactions on Programming Languages and System (TOPLAS)*, 23:1–29, jan 2001.
6. Iliano Cervesato, N. A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. A Meta-Notation for Protocol Analysis. In *Proc. CSFW*, pages 55–69, 1999.
7. D.E. Denning. A Lattice Model of Secure Information Flow. *Comm. of ACM*, 19(5):236–242, 1976.
8. D. Dubois, H. Fargier, and H. Prade. The Calculus of Fuzzy Restrictions as a Basis for Flexible Constraint Satisfaction. In *Proc. of IEEE International Conference on Fuzzy Systems*, pages 1131–1136. IEEE Press, 1993.
9. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of Bounded Security Protocols. In N. Heintze and E. Clarke, editors, *Proc. FMSP'99*, 1999.
10. E. C. Freuder and R. J. Wallace. Partial Constraint Satisfaction. *AI Journal*, 1992.
11. Evelyn Gray. American national standard t1.523-2001, telecom glossary 2000. published on the Web at http://www.its.bldrdoc.gov/projects/telecomglossary2000, 2001.
12. G. Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Information Processing Letters*, 56(3):131–133, 1995.
13. A.K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1):99–118, 1977.
14. U. Montanari. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Science*, 7:95–132, 1974. Also Technical Report, Carnegie Mellon University, 1971.
15. B. C. Neuman and T. Ts'o. Kerberos: An Authentication Service for Computer Networks, from IEEE Communications Magazine, September, 1994. In *William Stallings, Practical Cryptography for Data Internetworks*. IEEE Press, 1996.
16. Zs. Ruttkay. Fuzzy Constraint Satisfaction. In *Proc. of 3rd IEEE International Conference on Fuzzy Systems*, pages 1263–1268, 1994.
17. T. Schiex. Possibilistic Constraint Satisfaction Problems, or "How to Handle Soft Constraints?". In *Proc. of 8th Conference on Uncertainty in AI*, pages 269–275, 1992.
18. T. Schiex, H. Fargier, and G. Verfaille. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 631–637. Morgan Kaufmann, 1995.

Confidentiality Levels and Deliberate/Indeliberate Protocol Attacks (Transcript of Discussion)

Stefano Bistarelli

Istituto di Informatica e Telematica, C.N.R

Ernie Cohen: Let me defend classical protocol analysis and claim that what you propose is simply an abstraction of what has already been done.

In classical analysis we model various sorts of "out of band" security vulnerabilities, such as an agent exposing his long-term key, with `oops` actions (parameterized by various things like agent identity). When we prove a secrecy theorem, it usually comes with some caveats that arise from these `oops` actions. For example, we might prove that a session key negotiated between A and B remains secret unless either A or B has given away their long-term key. This "unless" part of the theorem is exactly a lower bound of your "security level" for the session key: simply replace logical `and`, `or` and `not` with your semiring `dot`, `plus` and `complement`, since classical analysis takes place in the initial boolean semiring. The extra complication buys us nothing.

Reply: I don't know if it already gives the precision that you want. For example, maybe you want to make some distinction between losing your credit card or office badge.

Ernie Cohen: You just model these with different `oops` actions and so would get different predicates corresponding to an attack that came through credit card loss and through the other type of attack. This is again classical analysis, so we don't say whether there is an attack or there isn't an attack, we just prove things that are as strong as possible. If the protocol is weak then we get a result that says something like this: either what you think happened actually happened, or else what actually happened was what this predicate describes, various combinations of `oops`es that could have brought about an attack. That gives of course the exact mechanism.

Reply: I think it depends how deep you want to go in the analysis.

James Malcolm: You said that you try and make the two constraint problems match, and if they don't match then there's a problem. Does your method produce any clues or hints that help you to see what the attack might be?

Reply: Yes. When I say that they match the problem, the slide is saying, "I compare, I compute the solution, I match the problem, and there is an attack." I only know there is an attack, I don't know where. But I can just concentrate on a part of the problem and compare that part, then if there is some trouble in that part I will find the reason there.

Analyzing Delegation Properties

Giampaolo Bella[1,2] and Lawrence C. Paulson[1]

[1] Computer Laboratory, University of Cambridge
15 JJ Thomson Avenue, Cambridge CB3 0FD, (UK)
{gb221,lcp}@cl.cam.ac.uk
[2] Dipartimento di Matematica e Informatica, Università di Catania
Viale A. Doria 6, I-95125 Catania (Italy)
giamp@dmi.unict.it

Abstract. Previous work on proving non-repudiation properties using the Inductive Method seems to be reusable for proving delegation properties. Current experiments with Crispo's delegation protocol support this claim. It follows that the two properties are closely related, though they are used in different contexts to convey different guarantees. It is expected that one man-month is required to develop full machine proofs.

1 Overview

Electronic Commerce should provide a number of safeguards if it is to attract more users. Identifying the other party and protecting our secrets during transmission are essential, but it is equally necessary to protect our interests against other principals' false claims. For example, someone might deny to have ordered certain goods. An eavesdropper might pretend that someone else had defrauded him.

Protection against false claims would come from the ability to fully trace the development of the electronic transactions. We call this property *traceability*. Precisely, the protocol sessions that carry out the transactions should be traced in order to pinpoint the evidence required to refute a false claim. If that is perhaps an ideal property, *non-repudiation* and *delegation* are realistic forms of traceability. The meaning of these properties, which will be discussed in the sequel, might be intuitive. But establishing whether a protocol guarantees them is not at all immediate, especially after the experiences with classical authentication protocols.

Specific protocols have been designed to achieve the new properties. The Zhou-Gollmann protocol [7] perhaps is the best known non-repudiation protocol. It protects pairs of principals from each other's denial of having participated in a session by sending or receiving a specific message. Each principal gets evidence that can be presented to a judge. A recent protocol for delegation is due to Crispo [4]. It allows a principal called *grantor* to delegate a set of rights/accountabilities to another principal called *grantee* (here, accountability stands for responsibility). The grantor requires evidence that he has transferred the accountabilities, so he is protected in case the grantee uses them to act

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 120–127, 2004.

illegally — the grantee should be held accountable. This property is called *delegation of accountabilities*. The grantee in turn seeks evidence that he has received the rights, so he is protected in case they are rights for illegal activity — the grantor commissioned that activity. This property is called *delegation of rights*.

We proved non-repudiation properties successfully [2] using the Inductive Method for protocol analysis [6], extended with a simple formalisation of agents' knowledge [1]. Those properties reduced to the fact that specific events precede others, and suitable proof strategies could be developed [3] for the purpose using the theorem prover Isabelle. We are currently studying delegation properties, and expect to be able to reuse the same proof strategies. We are analysing Crispo's protocol, and have already come to an inductive formalisation for it. Creation and transmission of messages could be specified routinely in terms of the existing `Says`, `Notes`, and `Gets` events. The only potential difficulty in the formalisation was to allow for the protocol use of several pairs of asymmetric keys. Each agent has a signature key pair, and the grantor also has a delegation key pair that he uses to delegate tasks. The grantee has a key pair to stipulate acceptance of a delegated task, and another pair to exercise that task.

We have developed a number of claims about the protocol, in the attempt to show that both the grantor and the grantee are safeguarded from each other's bad intent. The proofs of those claims are currently sketched on paper, and their Isabelle mechanisation is still under development. Difficulties arise from the management of the various key pairs, which require suitable lemmas for symbolic evaluation of expressions involving the `analz` operator (this operator recursively extracts atomic messages out of a set of messages, using the keys that become available). Also the presence of the spy, who can intercept and fake messages at will, complicates the proofs.

2 A Delegation Protocol

This section presents Crispo's delegation protocol [4] using the conventional "$A \to B : m$" notation, extended with a "$\downarrow A : m$" notation signifying that A notes down m.

1. $G \to g$: $G, g, \{\!|G, g, \Omega, DK_G|\!\}, \{\!|G, g, \Omega, DK_G|\!\}_{SK_G^{-1}}$
2. $g \to G$: $g, G, \{\!|g, \Omega, AK_g, EK_g|\!\}, \{\!|g, \Omega, AK_g, EK_g|\!\}_{SK_g^{-1}}$
3. $G \to g$: $G, g, \{\!|g, G, \Omega, DK_G, EK_g, AK_g|\!\}, \{\!|g, G, \Omega, DK_G, EK_g, AK_g|\!\}_{DK_G^{-1}}$
4. $\downarrow g$: $\{\!|g, G, \Omega, DK_G, EK_g, AK_g|\!\}, \{\!|g, G, \Omega, DK_G, EK_g, AK_g|\!\}_{DK_G^{-1}}$
   $\{\!|\{\!|g, G, \Omega, DK_G, EK_g, AK_g|\!\}, \{\!|g, G, \Omega, DK_G, EK_g, AK_g|\!\}_{DK_G^{-1}}|\!\}_{AK_g^{-1}}$

Fig. 1. Crispo's delegation protocol

The protocol steps are in Figure 1. The syntax for cryptographic keys is somewhat different from that used by the protocol author. Here, SK_X, DK_X, AK_X, and EK_X indicate principal X's public keys respectively for signature, delegation, acceptance, and exercise. The corresponding private keys are SK_X^{-1}, DK_X^{-1}, AK_X^{-1}, and EK_X^{-1}. It can be seen that some sub-messages are repeated. Precisely, each signed message carries aside its clear-text version for the sake of signature verification. The presentation could be shortened by introducing some abbreviations, but we believe that deriving a formalisation from this extended presentation is quicker.

The first step of the protocol sees the grantor G initiate a session with the grantee g by a four-component message. The first two contain the principals' identities. The third component is a message expressing G's wish to delegate to g accountability for a task Ω using DK_G. The fourth component is a copy of the third signed by G's private signature key. Clearly, the last two components constitute a digital signature, but perhaps the first two could just be derived from the underlying transport protocol.

In the second step, g replies with a message of the same structure. The digital signature, which uses g's private signature key, means that g accepts accountability for Ω using AK_g, and will exercise Ω using EK_g.

The third step lets G issue a confirmation message about the terms of the delegation. His signature by DK_G^{-1} is affixed on a message confirming that G delegated Ω to g using DK_G, and that g accepted Ω using AK_g and will exercise it using EK_g.

Finally, g extracts the digital signature derived from G's confirmation message, signs is by AK_g^{-1}, and notes down these two components, which form the *delegation token*. Whenever g wishes to exercise Ω, he must present the delegation token. Its internal signature by DK_G^{-1} confirms that it was the grantor who delegated the task.

3 Analysing a Delegation Protocol

This section reports on the state of our analysis of Crispo's delegation protocol. We have augmented the notation of the Inductive Method with intuitive syntax for the various types of asymmetric keys. For example, in the case of signature keys, `(pubSK X)` denotes SK_X and `(priSK X)` denotes SK_X^{-1}.

The protocol must be formalised as a set of traces of events, here `crispo`. The inductive definition of this set is in Figure 2, where three standard rules have been omitted: one stating the basis of the induction, one allowing for the spy's tampering, and another allowing for receipt of messages that have been sent. The figure only presents rules `C1`, ..., `C4`, which correspond to the protocol steps. Each of them is inductive, so it details how to extend a given trace of `crispo` with an extra event.

Rule `C1` formalises the first step. Defining the abbreviation `M` as a precondition makes the rule more compact. Notice that `M` contains `G`'s public delegation key. Rule `C2` specifies `g`'s reply, so it requires that `g` receive the message of the first

step. The form of that message is here specified by the abbreviation `M`. Another abbreviation, `M'`, makes the rule more compact. Rule `C3` has the same structure. Finally, rule `C4` introduces on the current trace the `Notes` event whereby `g` notes down the delegation token `T` along with its signature by `(priAK g)`.

```
C1:  "⟦ evs1 ∈ crispo;
        M = {|Agent G, Agent g, Number Ω, Key (pubDK G)|} ⟧
     ⟹ Says G g {|Agent G, Agent g, M, Crypt (priSK G) M|}
          # evs1 ∈ crispo"

C2:  "⟦ evs2 ∈ crispo;
        Gets g {|Agent G, Agent g, M, Crypt (priSK G) M |} ∈ set evs2;
        M  = {|Agent G, Agent g, Number Ω, Key (pubDK G)|};
        M' = {|Agent g, Number Ω, Key (pubAK g), Key (pubEK g)|} ⟧
     ⟹ Says g G {|Agent g, Agent G, M', Crypt (priSK g) M'|}
          # evs2 ∈ crispo"

C3:  "⟦ evs3 ∈ crispo;
        Gets G {|Agent g, Agent G, M', Crypt (priSK g) M'|} ∈ set evs3;
        M'  = {|Agent g, Number Ω, Key (pubAK g), Key (pubEK g)|};
        M'' = {|Agent g, Agent G, Number Ω,
                 Key (pubDK G), Key (pubAK g), Key (pubEK g)|} ⟧
     ⟹ Says G g {|Agent G, Agent g, M'', Crypt (priDK G) M''|}
          # evs3 ∈ crispo"

C4:  "⟦ evs4 ∈ crispo;
        Gets g {|Agent G, Agent g, M'', Crypt (priDK G) M''|}
          ∈ set evs4;
        M'' = {|Agent g, Agent G, Number Ω,
                 Key (pubDK G), Key (pubAK g), Key (pubEK g)|};
        T = {|M'', Crypt (priDK G) M''|} ⟧
     ⟹ Notes g {|T, Crypt (priAK g) T|} # evs4 ∈ crispo"
```

Fig. 2. Formalising Crispo's delegation protocol inductively

We have designed two claims formalising delegation of accountabilities, depending on whether the grantor `G` trusts the grantee `g` or not. The following claim concerns the former case, so it assumes `g ≠ Spy`.

```
⟦Crypt (priSK g) M' ∈ parts (knows G evs);
 M' = {|Agent g, Number Ω, Key (pubAK g), Key (pubEK g)|};
 g ≠ G; g ≠ Spy; evs ∈ crispo⟧
⟹ Says g G {|Agent g, Agent G, M', Crypt (priSK g) M'|} ∈ set evs
```

The first two assumptions signify that a specific message signed by `g`'s private signature key is part of `G`'s knowledge on a trace `evs` of the protocol model. When this is the case, the claim concludes that `g` indeed sent `G` an instance of message

2 containing that signature. So, by exhibiting a digital signature, `G` shows that `g` accepted accountability for Ω by `(pubAK g)` and will exercise it by `(pubEK g)`. The thread of the proof is that `G` can learn the signature either from receiving message 2, in case he is not the spy, or from overhearing that message as it is sent by `g`, in case he is the spy. The latter case leads directly to the conclusion of the claim. The former case requires a lemma stating that message 2 could be sent by `g` only, because the signing key is secure.

If we omit the assumption `g ≠ Spy`, then we cannot specify the form of the message that `g` sends, as can be read from the following claim.

```
⟦Crypt (priSK g) M' ∈ parts (knows G evs); g ≠ G; evs ∈ crispo⟧
⟹ ∃C Y. Says g C Y ∈ set evs
        ∧ Crypt (priSK g) M' ∈ parts {Y}
```

Intuitively, if `g` is the spy, then he can send messages of any form, but the claim says that any of his signatures must have originated with him. The proof thread resembles the previous one, but this time the event of the conclusion cannot be further specified.

The claims about delegation of rights are analogous, but are intended to defend `g`. The following one assumes `G ≠ Spy`. If `g` exhibits a specific message signed by `G`'s private delegation key, then `G` must have sent message 3 containing that signature. This confirms that `g` has the rights to exercise Ω.

```
⟦Crypt (priDK G) M'' ∈ parts (knows g evs);
 M'' = {|Agent g, Agent G, Number Ω,
         Key (pubDK G), Key (pubAK g), Key (pubEK g)|};
 g ≠ G; G ≠ Spy; evs ∈ crispo⟧
⟹ Says G g {|Agent G, Agent g, M'', Crypt (priDK G) M''|} ∈ set evs
```

By omitting the assumption `G ≠ Spy`, we come to the most general claim. Whenever `g` exhibits a message signed by `G`'s private delegation key, that signature originated with `G` inside a message of unspecified form.

```
⟦Crypt (priDK G) M'' ∈ parts (knows g evs); g ≠ G; evs ∈ crispo⟧
⟹ ∃C Y. Says G C Y ∈ set evs
        ∧ Crypt (priDK G) M'' ∈ parts {Y}
```

Additional guarantees can be developed about the delegation tokens. For example, it is important to prove that any token signed by a grantee's private acceptance key must have originated with the grantee, regardless his being the spy.

4 Conclusions

We are analysing the delegation protocol due to Crispo [4] using the Inductive Method. An inductive formalisation of the protocol is available, but the relevant

claims discussed above are still being proved. We expect that one man-month in total is sufficient to complete the entire mechanisation. Our guarantees are loosely related to those by Crispo and Ruffo [5], although the similarities still need to be examined.

We are confident that the same proof strategies that were previously used to prove non-repudiation properties can be reused here. Our proof attempts have highlighted that both non-repudiation and delegation properties reduce to proving that an event involving a principal precedes an event involving another principal. Such properties typically establish that a principal participated in a protocol session by either sending or receiving a message, on assumptions that concern another principal. Therefore, non-repudiation and delegation are related properties, although they are used in different contexts to convey different guarantees. It is fair to address both of them as forms of traceability.

To the best of our knowledge, this is the first contribution towards the mechanised analysis of delegation properties. We were pleased to observe once more that the Inductive Method is not limited to reasoning about authentication and confidentiality.

Acknowledgements. We thank Bruno Crispo for useful discussions about his protocol, and Giancarlo Ruffo for helpful comments on a draft of this paper. Research was funded by the EPSRC grant GR/R01156/01 *Verifying Electronic Commerce Protocols*.

References

1. G. Bella. Modelling Agents' Knowledge Inductively. In B. Christianson, B. Crispo, J. A. Malcolm, and R. Michael, editors, *Proc. of the 7th International Workshop on Security Protocols*, LNCS 1796, pages 85–94. Springer-Verlag, 1999.
2. G. Bella and L. C. Paulson. A Proof of Non-Repudiation. In *Proc. of the 9th International Workshop on Security Protocols*, LNCS Series. Springer-Verlag, 2001. In Press.
3. G. Bella and L. C. Paulson. Mechanical Proofs about a Non-Repudiation Protocol. In R. J. Boulton and P. B. Jackson, editors, *Proc. of the 14th International Conference on Theorem Proving in Higher Order Logics (TPHOLs'01)*, LNCS 2152, pages 91–104. Springer-Verlag, 2001.
4. B. Crispo. Delegation Protocols for Electronic Commerce. In *Proc. of the 6th Symposium on Computers and Communications (ISCC'01)*. IEEE Press, 2001.
5. B. Crispo and G. Ruffo. Reasoning about Accountability within Delegation. In *Proc. of the 3th International Conference on Information and Communications Security (ICICS'01)*, LNCS 2229. Springer-Verlag, 2001.
6. L. C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6:85–128, 1998.
7. J. Zhou and D. Gollmann. A Fair Non-Repudiation Protocol. In *Proc. of the 15th IEEE Symposium on Security and Privacy*, pages 55–61. IEEE Press, 1996.

Analyzing Delegation Properties (Transcript of Discussion)

Lawrence C. Paulson

University of Cambridge

Ernie Cohen: You could actually strengthen that last point. Basically, all the safety properties of these protocols are essentially the same, and on the whole these things only seem different because security people have, for some reason, considered authentication to be a different kind of thing.

Reply: Well I think you need to be fair and consider that at one level of abstraction we can model these by "this precedes that", but originally they arise from different goals, and have different meanings. If we can model them in this nice way, it's great, but I think the original security goals that people had in mind at a higher level were different notions. If they turn out all to be instantiations of the same concept, that's great, but I don't think it was obvious at the outset.

Bruce Christianson: But the thing being authenticated here is an event.

Ernie Cohen: That's simply because of the particular modelling that's used. If you just keep history variables then you're just authenticating some property of the history.

Bruce Christianson: Oh sure, you're authenticating properties of the state or of the history. You're using "authenticating" in a different way to the way that some of these earlier people did, they'd been using it to talk about authenticating people.

Ernie Cohen: Right, and that was the mistake. There's a general lesson here: if you're doing something with security, first look to see if somebody in ordinary vanilla distributed computing has already been doing the same thing; if they have then don't invent a new name for it.

Bruce Christianson: Perhaps a part that's not entirely obvious is that delegation is actually a more fundamental property than the authentication of an input.

Matt Blaze: This is a tangential question motivated by your example from the workflow of the system, in which Jill essentially needs to maintain proof that she was given the authority to do something that she may be called to account for later. In general, does it matter who maintains these messages? That is, in the workflow example, one can imagine that Jill is responsible for maintaining the delegation of authority that she received in the event of a future audit; and now you have the situation in which parts of the audit trail are maintained in different places. Potentially they are parties with very different interests. In particular, if Jill doesn't maintain the message that gives her the authority to delegate an authority to do something, it may be relatively straightforward to make it appear that she's done something wrong because Sam could erase it from wherever it's stored. On the other hand, if she is the one who's responsible for

maintaining it, then it may not be possible to audit all of the events in precisely the same place.

Reply: I would say that as a verification expert, that sort of thing is none of my concern; it is the concern of the company's security policy and so on.

Matt Blaze: Yes, I think this might be a practical problem with systems that depend on credentials in general: the policy for maintaining credentials related to transactions is quite important for the purposes of auditing compliance with the policy.

Reply: Whether or not I keep a receipt from a transaction at a shop, it doesn't affect the shop's own accounts of their transactions. I think this sort of thing can be treated in the same way.

Bruce Christianson: The sort of criterion which I guess Matt is proposing is that you should keep anything that it's in your interest to keep, and protocols should be designed in such a way that you can never have one party that can throw something away and cause a dispute that needs to be resolved against another party.

Matt Blaze: Yes, and as someone who routinely throws things that are attributed keys away, I'm very uncomfortable about protocols like that.

Bruce Christianson: I think part of the motivation for having these different keys is the recognition that the events we end up authenticating may happen on different pieces of hardware, and we don't want people communicating private keys between pieces of hardware. We talk about Jill as if she were a single entity, but that's actually misleading because it may not be Jill if she is a collection of different pieces of hardware. These pieces of hardware may not belong to, or be under the control of a single party. Jill may be a composite entity, she may be a corporation. She may have parts that don't trust each other, and the right to accept delegations may be itself something that has been delegated from someone else. That's why you have these different keys, and I think that actually makes it easier to split up the credentials into pieces that each person needs to keep. Provided you have a conflict resolution rule, that usually these protocols don't specify, there are usually other participants who say, "if you can't produce your credentials, the dispute's resolved against you and the penalty is . . . ". You need to have all those in the open to understand how that actually works.

Reply: Funny, I had a completely opposite solution which was if Alice or Jill or whoever belonged to the office workers union and they can maintain a storeroom to which you're meant to email all of your delegation tokens, they'll keep them safe for you.

Combinatorial Optimization of Countermeasures against Illegal Copying

Ryoichi Sasaki[1], Hiroshi Yoshiura[2], and Shinji Itoh[2]

[1] Tokyo Denki University,
2-2, Kanda-Nishiki-cho, Chiyoda-ku, Tokyo 101-8457, Japan
sasaki@im.dendai.ac.jp
[2] Hitachi, Ltd., Systems Development Laboratory,
292, Yoshida-cho, Totsuka-ku, Yokohama, 244-0817, Japan
{yoshiura,s-itoh}@sdl.hitachi.co.jp

Abstract. The growth of the Internet has lead to an increase in the distribution on digital contents, such as music, still pictures, and motion pictures through networks. However, it has also lead to an increase in illegal copying without permission from the copyright holder. Although a number of countermeasures against illegal copying have been developed including watermarking and contents capsulation, it is impossible to find one perfect countermeasure. A method is needed that would allow for an optimal combination of different countermeasures to substantially reduce the probability of illegal copying. We propose such a method based on combinatorial optimization and fault-tree analysis. We use this method for a number of cases and show the actual optimal combinations for each case, as well as a flow chart for selecting a proper combination of countermeasures depending on the situation.

1 Introduction

The growth of the Internet has resulted in a wide distribution of digital contents, such as music and motion pictures. Because digital contents are easy to copy without any degradation in quality, peer-to-peer file sharing systems such as Napster and Gnutella have been developed and have led to widespread copying without permission from the copyright holder. There is, therefore, a need for countermeasures against illegal copying such as watermarking or contents capsulation. Many studies have investigated digital watermarking [1] [2] and other techniques [6] to prevent illegal copying.

However, it is extremely difficult to find one perfect countermeasure against illegal copying by PC owners inside a firewall. A systems approach is required to arrive at an acceptable solution in terms of network security, cost, and so on.

In this paper, we propose a method based on combinatorial optimization and fault-tree analysis. We use this method for a number of cases and show the actual optimal combinations for each case, as well as a flow chart for selecting a proper combination of countermeasures depending on the situation.

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 128–144, 2004.

2 Countermeasures against Illegal Copying

2.1 Categories of Digital Contents Distribution Systems

There are many types of digital contents distribution systems depending on distribution media, treated media, computer systems etc. as follows:

(1) Distribution media:
  (a) Package media (e.g., CDs, DVDs),
  (b) Broadcasting media (e.g., satellite broadcasting)
  (c) Communication media (e.g., the Internet)
  In this paper, we focus on the Internet.
(2) Treated media
  (a) Music
  (b) Movies (Motion pictures)
  (c) Still Pictures
  (d) Text
  In this paper, we focus primarily on music and still pictures.
(3) Computer systems
  (a) Client server systems
    - Stored service - Streaming service
  (b) Peer-to-peer systems
  This paper focuses on the stored service in client server systems
(4) Replay equipment in client server systems
  (a) General PCs
  (b) Special equipment (e.g., mobile phones, DVD-RAM players)
  This paper focuses on the above two types of equipment

2.2 Methods of Illegal Copying

There are many definitions of " illegal copying" and there are many ways of producing illegal copies. In this paper, we focus on the following two ways of obtaining illegal copies.

(1) Illegal copying by illegally accessing contents distributors authorized by the copyright holder.
(2) Illegal copying by copying the contents uploaded illegally to the server without permission from the copyright holder.

2.3 Measures against Illegal Copying

With respect to technology, there are two types of measures against illegal copying (See Fig. 1):

(1) Measures to protect contents against illegal copying, which means direct countermeasures against illegal copying.
(2) Measures to prevent illegal copying, which means measures that reduce the number of attempts to make illegal copies.

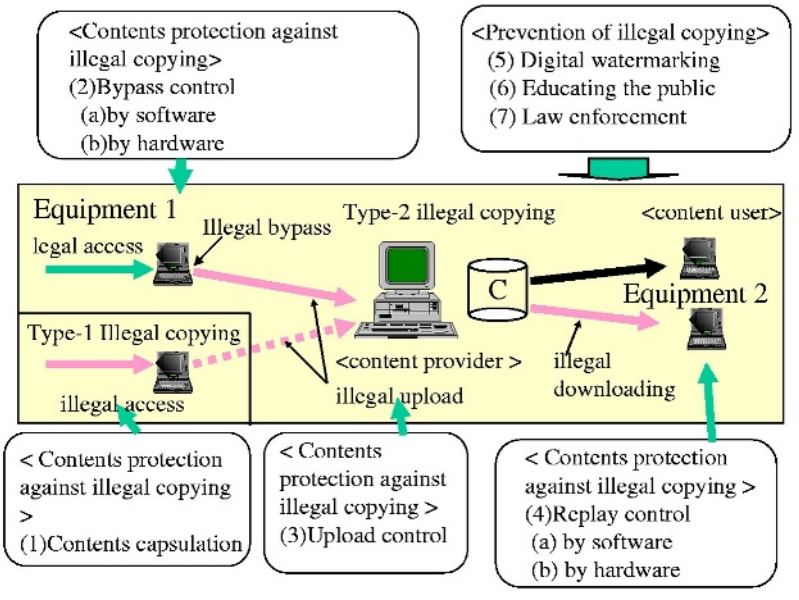


Fig. 1. Countermeasures.

Measures to protect contents against illegal copying. Measures to protect contents against illegal copying can be categorized as shown in Fig. 1.

(1) Contents capsulation on the Internet
(2) Bypass control on PCs or special equipment, e.g., mobile phones
(3) Uploading control at the distribution server
(4) Replay control on PCs or special equipment, e.g., mobile phones

We will now describe each countermeasure in some detail.

(1) Contents capsulation on the Internet

Fig. 2 illustrates a simple method of contents capsulation for copy protection in the case of mobile phones. This method is also applicable to PCs. As shown here, the mobile phone in which the license key is stored will receive encrypted music contents from the music delivery server. Only the mobile phone that has the proper license key can play the music after decrypting the encrypted contents.

Someone who copies encrypted music contents from a flash card cannot listen to the original music without the proper license key. Because this key is stored in special tamper-resistant equipment, it is inaccessible to anyone except a professional pirate. The decrypted music contents inside the mobile phone can be stolen, but only by someone skilled enough to access special tamper-resistant equipment. There are, of course, many variations of the above method that can reduce the number of required encryptions and can make it more useful [3]. This measure is useful for protecting the contents against illegal copying by illegally accessing contents distributors.

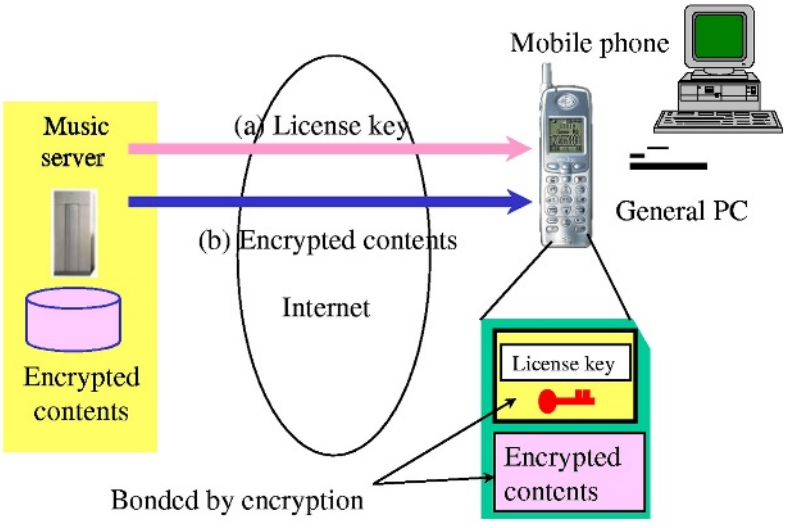


Fig. 2. Contents capsulation.

(2) Bypass control on PCs or special equipment

Fig. 3 shows a bypass control tool. Software and/or hardware controls the bypass to prohibited devices such as flexible discs. Although in this case, it is possible to replay the contents, it is impossible to copy the contents on the disc. This measure is useful for protecting the contents against illegal copying by copying the contents uploaded illegally to the server without permission from the copyright holder.

(a) Examples of bypass control by hardware
The equipment has no output device for copying. This feature is useful for special equipment such as mobile phones, however it is impossible to use this feature for general PCs.
The copy function on the hardware or firmware does not work if the contents contain restricted information, even if the "copy" command is given to the equipment.

(b) Examples of bypass control by software
The copy function on the software does not work when prohibiting marks are embedded in the contents, even if the "copy" command is given to the equipment.

(3) Uploading control at the distribution server

Providers refuse to upload the contents without permission from the copyright holder. Providers can also use a system for the automatic rejection of requests to upload contents. This measure is useful for protecting the contents against illegal copying by copying the contents uploaded illegally to the server without permission from the copyright holder.

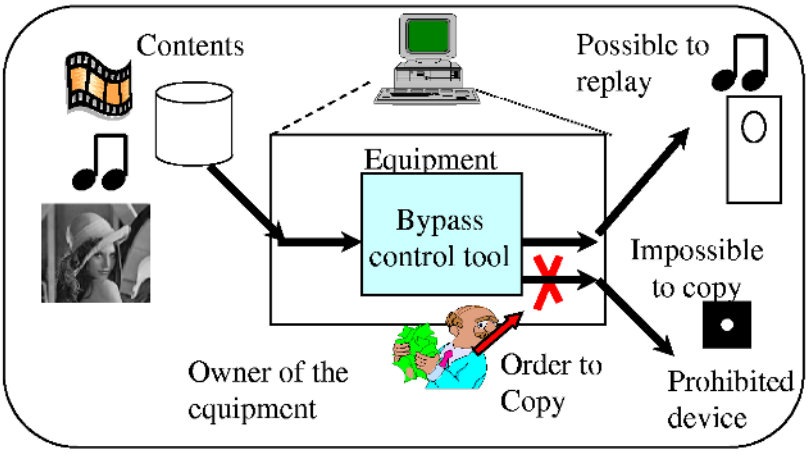


Fig. 3. Image of bypass control.

(4) Replay control on- PCs or other equipment

Software or hardware controls the replay of contents when there are prohibiting marks embedded in the contents, even if the replay command is given to the equipment. This measure is useful for protecting the contents against illegal copying by copying the contents uploaded illegally to the server without permission from the copyright holder.

Measures to prevent illegal copying. The following measures can be used to prevent illegal copying as shown in Fig.1.

(5) Digital watermarking
(6) Educating the public
(7) Law enforcement

The most widely used measure of illegal-copying prevention is digital watermarking [1] [2]. Information about the identity of the owner and/or buyer of the contents is embedded into the digital contents after it is converted into 0-1-bit patterns. For example, a pixel related to a particular bit will brighten if the bit value is 0 and darken if the bit value is 1.

If the watermarking is done properly, few people will be able to even see the watermarks embedded in the contents. Software for extracting the watermarks can be used to identify the owner and/or buyer. The owner can stop the business with the buyer identified by watermark extraction. This can directly reduce the probability of illegal copying.

3 Method for Obtaining the Optimal Combination of Countermeasures

3.1 General Procedure

We propose a general procedure to obtain the optimal combination of countermeasures as described below (see Fig. 4).

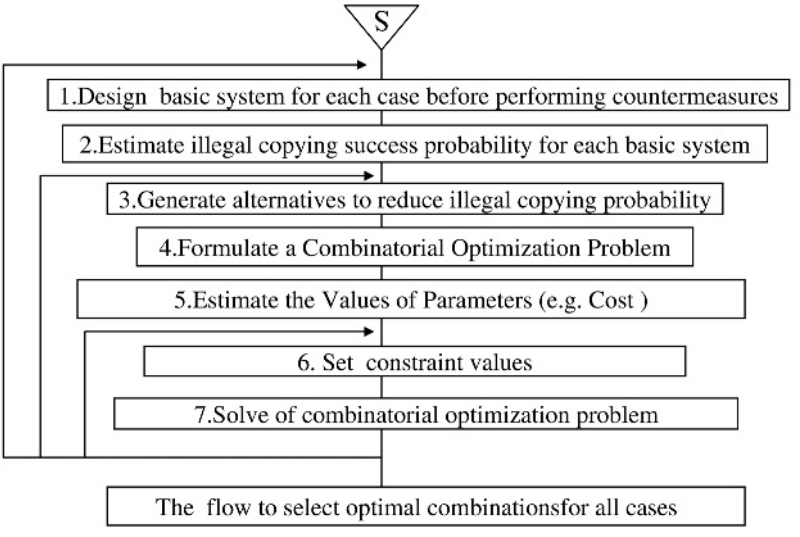


Fig. 4. Procedure to obtain optimized Countermeasures against illegal copying.

Step 1: Design a basic contents distribution system before using specific countermeasures. The design depends on the treated contents, e.g., music or still pictures as well as on the replay equipment, e.g., a general PC or some special equipment such as a mobile phone.

Step 2: Estimate the illegal-copying success probability for each basic system. We propose to use fault-tree analysis [5] for this estimation. Fig. 5 and Fig. 6 illustrate an example of fault trees.

Step 3: Generate alternative countermeasures to reduce the probability of illegal copying such as digital watermarking, contents capsulation, bypass protection, and so on as shown in Chapter 2.

Step 4: Formulate a combinatorial optimization problem. This formulation is discussed in detail in Section 3.2. Mathematically, this problem is represented as an integer programming problem [4].

Step 5: Estimate the value of each coefficient described in Section 3.2.

Step 6: Set the values of constraints such as the upper limit for the total cost as described in Section 3.2 in each situation.

Step 7: Obtain the optimal combination of countermeasures by solving the integer programming problem described in Section 3.2.

Steps 1 to 3 are not difficult. In addition, using a brute force method or a branch-and-bound method [4], we can easily solve the integer programming problem in Step 7

However, in Step 5, it is very difficult and time consuming to estimate the exact value of each coefficient such as income, cost, and loss caused by attacks in various situations. On the other hand, roughly estimating the value of each coefficient for limited alternative countermeasures is not very difficult. Rough estimation is acceptable in many cases to obtain a satisfactory solution.

3.2 Formulation of Combinatorial Optimization Problems

<Problem 1 >

Maximization

$$W \cdot T(1 - P(X_1, X_2, \cdots, X_n)) + V_0 - \alpha(\sum_{i=1}^{n} C_i(T) \cdot X_i - C_0) \tag{1}$$

subject to

$$\sum_{i=1}^{n} C_i(T) \cdot X_i C_T \tag{2}$$

$$X_i = 0 \text{ or } 1 (i = 1, 2, \cdots, n) \tag{3}$$

where

W is the unit price per copy (in yen),

T is the total number of copies, and

P is the function used to calculate the illegal-copying success probability after performing countermeasures. This probability is obtained by using fault tree analysis as shown in Fig 5 and Fig. 6. If the minimal cut sets are $< a, cde, cdf, b, nopjk >$, then the probability before performing countermeasures, P_T, can be calculated as follows.

$$P_T = 1 - (1 - P_a)(1 - P_c P_d P_e)(1 - P_c P_d P_f)(1 - P_b)(1 - P_n P_0 P_p P_j P_k) \tag{4}$$

If alternative i is implemented for event d, the probability after implementing alternative i can be calculated as $P_d \cdot (1 - \Delta P_i)$, where ΔP_i is the reduction in the illegal-copying success probability. This representation can be extended as follows to match both cases where alternative i is adopted, and those where it is not adopted.

$$P_d \cdot (1 - \Delta P_i \cdot X_i)$$

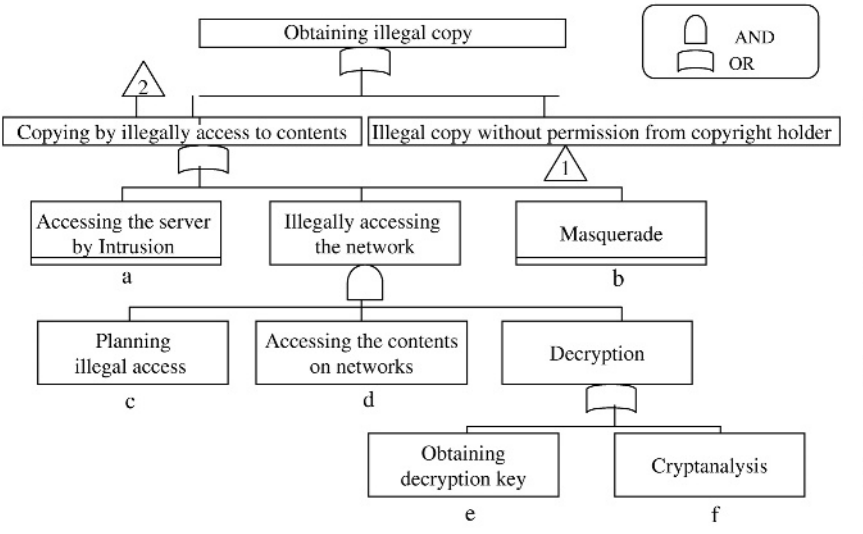


Fig. 5. Fault Tree Structure for Obtaining Probability of Illegal Copying (1).

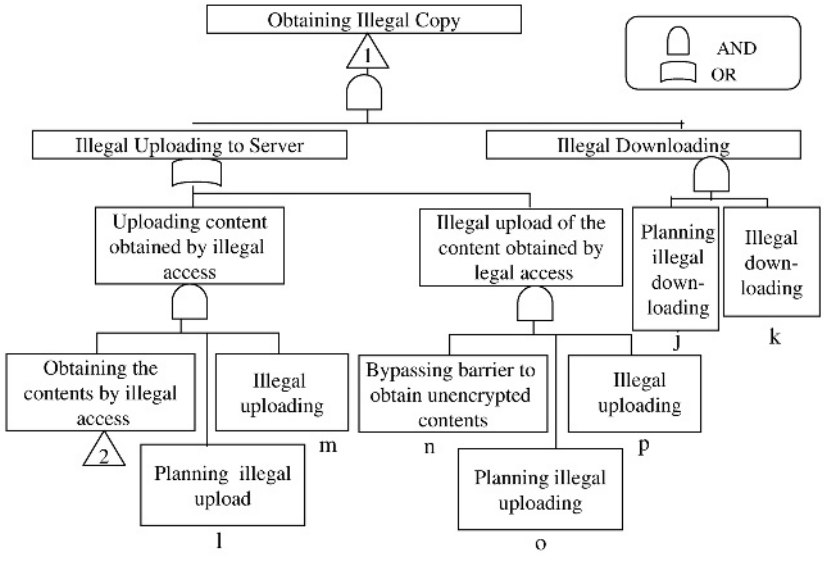


Fig. 6. Fault Tree Structure for Obtaining Probability of Illegal Copying (2).

When alternative countermeasures i and j are implemented for the same event, the function is represented as

$$P_d \cdot (1 - (1 - \Delta P_i \cdot X_j) \cdot (1 - \Delta P_j \cdot X_j)).$$

As shown above, $P()$ can be represented as the function of $X_i (i = 1, 2, \cdots, n)$.

C_i : the cost of implementing alternative i (in yen). This value is represented as the function of the total number of copies.
V_0 : the income without selling the contents (in yen).
C_0 : the cost without implementing alternative countermeasures (in yen).
α : the coefficient. If α is 1, equation (1) is a profit maximization problem. If α is zero, then it is an income maximization problem.
C_T : the value of the upper constraint on the implementation of countermeasures (in yen).

The formulation described above is just one example; other formulations are acceptable.

4 Results

4.1 Example of the Use of the Proposed Method

Step 1: Design a basic contents distribution system before using specific countermeasures.

As illustrated in Fig. 2, the basic system has a music server, replay, and the Internet. Music contents are encrypted outside the replay equipment by using a contents capsulation technique.

(Case 1) The music contents are downloaded from the server to a general PC via the Internet.
  (Case 1-1) The contents are free of charge
  (Case 1-2) The price of the contents is 500 yen.
(Case 2) The music contents are downloaded from the server to a mobile phone via a wireless telephone line and the Internet.

Step 2: Estimate the illegal-copying success probability for each basic system.

We used fault-tree analysis [5] for this estimation. The fault tree structure developed by the analysis is shown in Fig. 5 and Fig. 6. In this case, the minimal cut sets [5] are $< a, cde, cdf, b, nopjk >$. By using the minimal cut sets and the values of $P_a = 0$, $P_b = 0$, $P_c = 0.5$, $P_d = 0.1$, $P_e = 0$, $P_f = 0$, $P_j = 0.5$, $P_k = 1$, $P_l = 0.5$, $P_m = 1$, $P_n = 1$, $Po = 0.2$, and $P_p = 1$, we estimated the probability to be 0.1.

Step 3: Generate alternative countermeasures to reduce the probability of illegal copying.

We generated the following six alternatives (see Fig. 7).

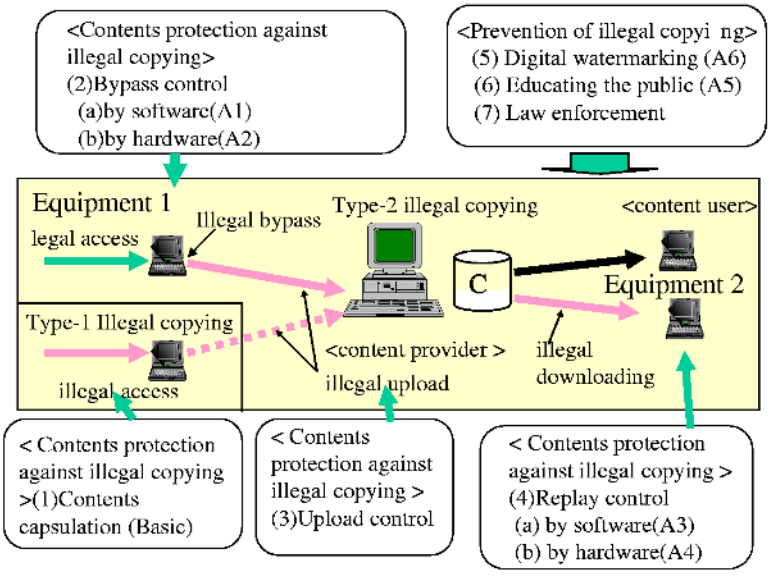


Fig. 7. Alternatives of Countermeasures.

Alternative 1 Bypass control by software to prohibited output devices in replay equipment 1.
Alternative 2 Bypass control by hardware to prohibited output devices in replay equipment 1.
Alternative 3 Replay control by software in equipment 2.
Alternative 4 Replay control by hardware in equipment 2.
Alternative 5 Educating the public to prevent illegal uploading of contents to a distribution server such as a Web server.
Alternative 6 Digital watermarking to find those who illegally upload contents.

Step 4: Formulate a combinatorial optimization problem.

This was formulated in the same way as Problem 1 in Section 3.2. In this case,

$$P(X_1, X_2, \cdots, X_n) = 1 - (1 - P_a)(1 - P_c \cdot P_d \cdot P_e)(1 - P_c \cdot P_d \cdot P_f)(1 - P_b)$$
$$\cdot (1 - P_n \cdot (1 - \Delta P_1 \cdot X_1) \cdot (1 - \Delta P_2 \cdot X_2) \cdot P_o \cdot (1 - \Delta P_3 \cdot X_3)$$
$$\cdot (1 - \Delta P_4 \cdot X_4) P_p \cdot P_j \cdot P_k \cdot (1 - \Delta P_5 \cdot X_5) \cdot (1 - \Delta P_6 \cdot X_6)) \qquad (4')$$

Step 5: Estimate the value of each coefficient described in Section 3.2.

For the actual situation we examined, we selected the values described in Table 1.

Table 1. Values of Parameters.

Alternative		Case	C_i	ΔP_i	Note
A1: Bypass control by software in equipment 1	n	1-1	$5000000 + T$	0.7	
		1-2	$5000000 + T$	0.7	Case 1-1
		2	$5000000 + T$	0.7	W=0 yen
A2: Bypass control by hardware in equipment 1	n	1-1	$500 \times T$	0.9	Case 1-2
		1-2	$500 \times T$	0.9	W=500 yen
		2	$50 \times T$	0.99	Case 2
A3: Replay control by software in equipment 2	k	1-1	$5000000 + T$	0.8	W=500 yen
		1-2	$5000000 + T$	0.8	
		2	$5000000 + T$	0.8	V_o = 1M yen
A4: Replay control by hardware in equipment 2	k	1-1	$500 \times T$	0.99	C_o = 2M yen
		1-2	$500 \times T$	0.99	
		2	$50 \times T$	0.999	Standard
A5: Educating the public to prevent illegal uploading	o	1-1	$10 \times T$	0.2	T = 50000
		1-2	$10 \times T$	0.2	C_T = 1M yen
		2	$10 \times T$	0.2	5M yen
A6: Digital watermarking embedded in contents	o	1-1	$10000000 + T$	0.3	10M yen
		1-2	$10000000 + T$	0.3	
		2	$10000000 + T$	0.3	

$P_a = 0$, $P_b = 0$, $P_c = 0.5$, $P_d = 0.1$, $P_e = 0$, $P_f = 0$, $P_j = 0.5$, $P_k = 1$, $P_l = 0.5$, $P_m = 1$, $P_n = 1$, $P_o = 0.2$, $P_p = 1$

(Cost estimation)

(a) In alternatives 1 and 3, we assumed that the software development cost for a PC or a mobile phone is 5M yen, and the settlement cost for one PC or a mobile phone is 10 yen. We also assumed that ten copes are made per PC or mobile phone. In alternative 6, the cost was estimated to be 10M yen, because not only digital watermarking software but also software for patrolling against illegal copying of the contents had to be developed.
(b) In alternatives 2 and 4, the hardware cost for copy protection on a PC was assumed to be 5000 yen per PC, because an additional kit had to be prepared. On the other hand, the cost of hardware for copy protection on a mobile phone was assumed to be 500 yen per mobile phone, because the protection function was embedded in the LSI together with other functions.
(c) The cost for educating the public was estimated to be 100 yen per person. One person was assumed to be making ten copies.

(Probability estimation)

(a) As mentioned earlier, $P_a = 0$, $P_b = 0$, $P_c = 0.5$, $P_d = 0.1$, $P_e = 0$, $P_f = 0$, $P_j = 0.5$, $Pk = 1$, $P_l = 0.5$, $P_m = 1$, $P_n = 1$, $P_o = 0.2$, and $P_p = 1$ were used to estimate the probability before performing countermeasures.
(b) The value of ΔP_i , which shows the effectiveness of the i-th countermeasure in reducing the illegal-copying probability was set under the following assumptions.

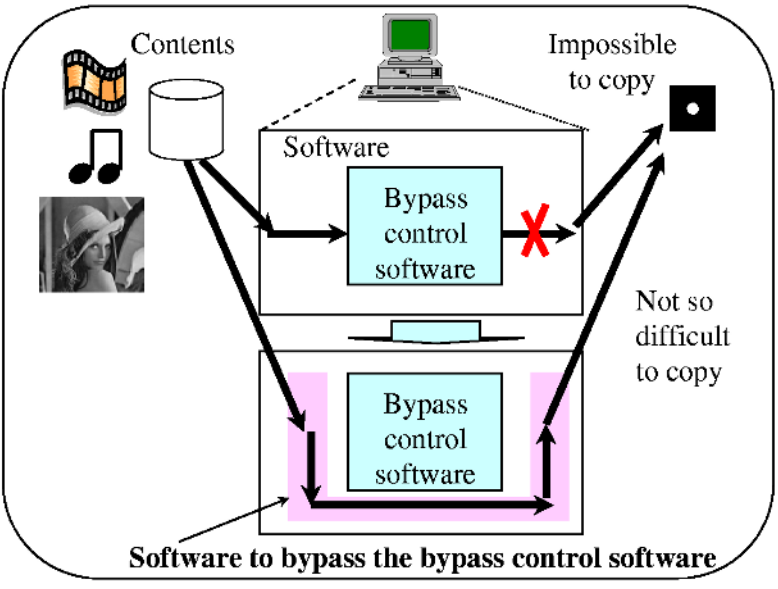


Fig. 8. Bypassing Copy protection.

The effectiveness of protective countermeasures is greater than that of preventive countermeasures.
The effectiveness of the hardware protection is greater than that of the software protection.

As shown in Fig. 8, software professional who is the owner of the general PC can easily develop the software to break bypass control by software to prohibited devices on a general PC. Therefore, we assumed that the effectiveness of the bypass control by software is small.

(others)

(a) As the standard case, 50000 was used for T that means the total number of copies made.
(b) $\alpha = 1$ was used in the standard case. This means that the objective function of this problem is profit maximization.

Step 6: Set the values of constraints such as the upper limit for the total cost as described in Section 3.2 in each situation. CT=1M, 5M, and 10M yen were used as constraint values.

Step 7: Obtain the optimal combination of countermeasures by solving the integer programming problem described in Section 3.2. A computer program based on a brute force method was developed and used to obtain the optimal solution and optimal value.

4.2 Results Obtained for the Example

Results in the standard case. Optimal solutions maximizing the profit in the standard case are shown in Table 2.

Table 2. Results of Computations (1).

Case	Item	Constraint Value C_T			Note
		1M Yen	5M Yen	10M Yen	
1-1	Optimal Value*	-1,000,000	-1,000,000	-1,000,000	
	OS**(x_1, x_2, $\cdots$)	000000	000000	000000	T=50000
1-2	Optimal Value*	21,500,000	21,500,000	21,500,000	$\alpha = 1$
	OS**(x_1, x_2, $\cdots$)	000010	000010	000010	(Profit
2	Optimal Value*	24,000,000	25,999,975	25,999,980	maximizaiton
	OS**(x_1, x_2, $\cdots$)	000010	000010	000010	Problem)

**Yen, **OS: Optimal Solution

(Case 1-1: The case in which a PC is used as client equipment, and the contents are free in charge)

In this case, of course, the optimal solution is not to use any measures. The profit (or debt) is obtained as V0-C0. New income without selling the digital contents such as that from advertising is required to enlarge the profit.

(Case 1-2: The case in which a PC is used as client equipment, and the price of the contents is 500 yen)

In the standard case, the optimal solution is $X_1 = 0$, $X_2 = 0$, $X_3 = 0$, $X_4 = 0$, $X_5 = 1$, and $X_6 = 0$ regardless of the value of constraints. This solution means that in the standard case, you should use alternative 5, which means "Educating the public to prevent illegal uploading".

(Case 2: The case in which a mobile phone is used as client equipment, and the price of the contents is 500 yen)

When the value of constraints is less than 5M yen in the standard case, you should use alternative 5, which means "Educating the public to prevent illegal uploading", as in Case 1-2. If you can spend 10M yen on countermeasures, the optimal solution is $X_1 = 0$, $X_2 = 1$, $X_3 = 0$, $X_4 = 1$, $X_5 = 1$, and $X_6 = 0$. In this case, you should add alternative 2, which represents hardware protection of the bypass to prohibited output devices in replay equipment 1,and alternative 4, which means hardware protection of the bypass to prohibited output devices in replay equipment 2.

Results in the case of income maximization. The optimal solutions obtained in the case of maximizing the income are shown in Table 3.

(1) When the constraint on the countermeasure cost is strict, you should use alternative 5.

Table 3. Results of Computations (2).

Case	Total number of Copying	Constraint Value (C_T)			
		1M Yen	5M Yen	10M Yen	50M Yen
Case 1-2 (General PC)	50000	24,000,000	24,000,000	25,600,000	25,999,160
		000010	000010	001010	101111
	500000	226,000,000	231,000,000	231,000,000	250,160,000
		000000	000010	000010	101011
	5000000	2,251,000,000	2,251,000,000	2,451,000,000	2,490,500,000
		000000	000000	001000	101001
Case 2 (Mobile Phone)	50000	24,000,000	25,997,500	25,998,000	25,999,916
		000010	010100	010110	111111
	500000	226,000,000	231,000,000	246,000,000	250,999,750
		000000	000010	001000	010100
	5000000	2,251,000,000	2,251,000,000	2,451,000,000	2,490,500,000
		000000	000000	001000	010110

Optimal Value
Optimal Solution
$\alpha = 0$ (Income Maximization Problem)

(2) When a PC is used as client equipment for replaying the contents and the value of constraints is less than 10M yen, you should use alternatives 3 and 5 if the total number of copies is 50000. If you can spend around 50M yen on countermeasures, you should use alternatives 1, 3, 4, 5,and 6. When a PC is used, the software-based alternatives are generally preferable.
(3) When a mobile phone is used as client equipment for replaying the contents and the value of constraints is 5M yen, alternatives 2 and 4 are preferable. When the total number of copies increases, an alternative with which the cost increases in proportion to the total number of copies (such as alternative 5) should be chosen.

Flowchart for selecting a proper combination of countermeasures depending on the situation. From the above results, Fig. 9, which shows below can be obtained.

(1) If it is possible to make contents available free of charge, you should allow copying. This is because, generally speaking, copy protection is extremely difficult and expensive. You should find new business models to generate money without selling the contents.
(2) If special equipment such as a mobile phone is used for replaying the contents, hardware protection against the bypass to prohibited devices is useful. In this case, you should opt for copy protection.
(3) If a general PC is used and the available resources for countermeasures are not enough, you should choose preventive measures including contents capsulation and public education. Software professionals can easily break bypass

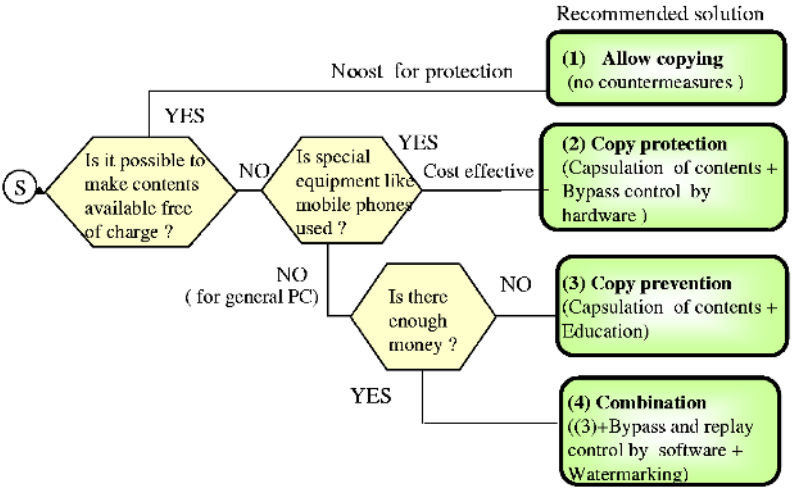


Fig. 9. Flow Chart for Selecting Proper Countermeasures against Illegal Copying.

control by software to prohibited devices on a general PC. In addition, average computer users can easily bypass the protection, because the tools developed by software professionals will be immediately uploaded to WEB sites.

(4) If a general PC is used and the money available for countermeasures is enough, you should choose a combination of countermeasures including the countermeasures described in (3), bypass and replay control by software to prohibited devices and digital watermarking.

5 Conclusions

We proposed a method to obtain the optimal combination of countermeasures against illegal copying based on combinatorial optimization and fault-tree analysis. We used it for a number of cases and showed the actual optimal combinations of countermeasures against illegal copying in each case, as well as a flow chart for selecting a proper combination of countermeasures depending on the situation.

We believe that the flow chart shown in Fig. 9 is useful for contents providers to roughly estimate, without much effort, what system of protection against illegal copying is preferable in a given situation.

References

1. Kineo Matsui, Fundamentals of Digital Watermarking, Morikitasyuppan, 1998 (in Japanese).

2. Stephan Katzenbeisser and A. P. Petitcolas (editors), Information Hiding Techniques for Steganography and Digital Watermarking, Artech House, 2000.
3. Ryoichi Mori and Masaji Kawahara, Superdistribution: the concept and the architecture, Trans. of the IEICE, vol. 73, no.7, pp. 1133–1146, 1990.
4. R. S. Gerfinkel and G. L. Nemhauser, Integer Programming ,Wiley, 1972.
5. N. J. McCormic, Reliability and Risk Analysis , Academic Press, 1981
6. Norihiko Sakurai et al. Trend on copy rights protection techniques for content distribution, Information Processing Society in Japan, Vol. 42, No. SIG 15 (TOD 12) pp. 63–76 December, 2001(Japanese)

Combinatorial Optimization of Countermeasures against Illegal Copying (Transcript of Discussion)

Ryoichi Sasaki

Tokyo Day University

Matt Blaze: One of the problems with digital rights management systems and copy prevention mechanisms in general, is that they assume that the policy is whatever the originator of the content wants it to be. Usually the originator of the content wants a policy that says, "no copying without my permission". That is all well and good from their perspective, but in fact it is often quite contrary to the laws of most countries with the rights a recipient may have. For example, in the United States, there is the concept of *fair use* that says that certain copying is permitted whether the originator of it wants it or not, and rights management systems often prevent these kinds of fair use rights. Does your model take into account the possibility that the owner of the content may not, in fact, be the setter of the policy?

Reply: This application was performed on the basis of the former case where the copyright holder has a strong right, but I think it would be possible for us to make an analysis of a model that includes fair use.

Laurent Eschenauer: I think you need to integrate market demand into your approach, because when you implement an alternative copy protection mechanism, some people won't buy the content because it is protected. The number of copies is dependent on the alternative that you implement, so this has to be integrated into the model.

Richard Clayton: One of the other strategies that right holders are using is to spend money on lawyers, either to do prosecutions on people who are doing copying, or to go and lobby people who are making the laws, in order to change things. It would be interesting to feed that into your model to see what the optimum amount to spend on lawyers would be! [Laughter]

Protocols with Certified-Transfer Servers

Raphael Yahalom

Onaro Research*
yahalom@onaro.com

1 Introduction

In some circumstances a party Q may have an incentive to perform *Selective Denial of Service* or *Selective Delay of Service* to other counter-parties in a protocol. That is, Q may use information at his disposal in order to selectively discriminate against certain service requests, for his own advantage. Q's counter-parties may not necessarily be in a position to prove such malicious behavior by Q.

For example, an on-line trader Q may have an incentive to delay the execution of certain trade requests and try to exploit, to his own advantage, the time-window between the request-arrival point and the actual request-execution point (selected to be optimal for Q). An on-line access-control server Q may have an incentive to delay performing certain key-revocation requests in order to postpone undesirable liability transfers. An on-line elections administration server Q may have an incentive to attempt to deny particular eligible voters from submitting their secret ballots in time, based on Q's assessment of the likely votes of particular demographic groups.

Such exploitation of information asymmetry represents a class of economic moral hazard problems [10]. In all these examples and others, Q may rely on the fact that it is impossible to prove whether, or exactly when, a particular request was actually *received* by Q. Q may exploit this inherent uncertainty in order to rely on his knowledge of the request to decide unilaterally whether, and when, to execute it.

In some cases, even if Q does not have malicious intent, he may not have sufficiently strong incentives to provide the high-availability and denial-of-service robustness which appropriately reflects the level of risk for his clients.

We propose an approach for dealing with these challenges which relies on a new 3rd-party *Certified-Transfer Service(CTS)*. In general, a CTS is a highly-available, highly-reliable service which simply accepts a message-blob from anyone, appends its timestamp and digitally-signs it, and posts the signed time-stamped message-blobs using a specified index - making the signed message-blob highly-accessible.

With a CTS, Q may thus sign a-priori a commitment stating that if a message with particular contents will be posted at the CTS by a certain time, he will take certain actions within a specified period of time.

* The author is performing part of this research work at MIT - Sloan School of Management

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 145–157, 2004.

P, or a delegate on her behalf, may verify Q's correct reactions, and, if required, rely on Q's a-priori commitment in conjunction with the CTS's signed statement to *provably-incriminate* Q in case of a violation.

The risks to the clients of selective denial of service (as well as of some forms of conventional denial of service) significantly diminish.

The two fundamental principles in the CTS design approach is objectivity and high-availability.

Objectivity is strengthened by the fact that the CTS gains no useful knowledge about the contents of messages, the application protocols with which they are associated, or the actual identity of the sender or the recipient. Consequently CTS has no incentive to perform malicious violations (such as itself performing a selective denial of service).

High-availability is strengthened by scalable, wide-scale replication of relatively non-complex servers .

The actual details of the design of the distributed CTS, based on secure and high-availability replication techniques, will be presented elsewhere.

Particular aspects of the CTS design benefit from important contributions made in Anderson's Eternity Service work [1], in Micali's Certified Post Office work [9], as well as in the body of work in the areas of Byzantine Generals protocols (e.g. [7]), Electronic Contract Signing (e.g. [2]), and Electronic Notary Services (e.g. [11]).

In this paper we discuss and demonstrate some subtle issues involved with the design of application protocols which rely on the CTS.

The main new challenge is associated with a trade-off between the need to reveal, in some cases, confidential information hidden in certain message components in order to enforce liability, and, on the other hand, the need to maintain the basic integrity and confidentiality guarantees of the protocol.

In this paper, we demonstrate this by enhancing an on-line elections protocol to rely on the CTS to prevent potential selective denial of service.

2 Certified-Transfer Service

In our notation $\{\ msg\ \}_{K_{P+}}$ represent the public-key encryption of msg with party P's public key, and $\{\ msg\ \}_{K_{Q-}}$ represents msg digitally signed with Q's private-key. A comma $''\ ,\ ''$ denotes concatenation between message components.

The design of a *Certified-Transfer Service* CTS is based on multiple distributed replicated servers to provide the following single service abstraction:

- Any party P can submit (possibly anonymously) to CTS a message encrypted with CTS's public key which contains an identifier and a message-blob (itself possibly encrypted with the key of some intended recipient):

  $$P\ \rightarrow\ CTS\textbf{:}\ \{\ l\ ,\ s_{id}\ ,\ [msg.blob]\ \}_{K_{CTS+}}$$

  where l is a location index, s_{id} is a unique submission identifier selected by P.

– Upon receipt of each submission, CTS generates, signs, and posts:

$$\{\ l\ ,\ s_{id}\ ,\ ts_{cts}\ ,\ s_{id.cts}\ ,\ [msg.blob]\ \}_{K_{CTS-}}$$

where ts_{cts} is CTS's receipt timestamp and $s_{id.cts}$ is a unique submission identifier assigned by CTS.
– Anyone can retrieve any of the signed posted entries from CTS [1];
– No one can delete or corrupt any of the signed posted entries from CTS (though they may be expired and moved off-line after period of time, determined by CTS's general policy).

Parties can rely on the CTS for reducing risks in their protocol executions. For example a participant server Q may be required to digitally sign a-priori a protocol contractual commitment statement prior to any actual execution of a protocol pr, along the lines of:

"*...for any message encrypted with my public-key "* K_{Q+} *", which include "* pr *" in a protocol identifier field, and which is submitted with an index "* l *" and id "* s *" at CTS at time "* ts *", I will submit a correct response according to the specification of "* pr *" which will be posted by CTS with an id "* s' *" at index "* l *" not after "* $ts + \delta$ *" ...*"

In a particular execution a client participant P may send the following submission to CTS:

$$P\ \rightarrow\ CTS\textbf{:}\ \{\ l\ , s_{id}\ ,\ [\{\ pr\ ,\ protocol.msg1\}_{K_{Q+}}]\ \}_{K_{CTS+}}$$

where s_{id} is a unique identifier index selected by P , $protocol.msg1$ is structured consistently with the specification of protocol pr (which is also possibly digitally signed by Q or contains other authentication components).

CTS posts the following message (which Q will look for via index l and retrieve):

$$\{\ l\ ,\ s_{id}\ ,\ ts_{cts}\ ,\ s_{id.cts}\ ,\ [\{\ pr\ ,\ protocol.msg1\}_{K_{Q+}}]\ \}_{K_{CTS-}}$$

Three outcomes are now possible:

1. P will retrieve Q's response via a CTS entry
$\{\ l\ ,\ s'_{id}\ ,\ ts'_{cts}\ ,\ s'_{id.cts}\ ,\ [\{\ pr\ ,\ [protocol.msg2]\}_{K_{P+}}]\ \}_{K_{CTS-}}$

   in which $ts'_{cts} \leq ts_{cts} + \delta$ and $protocol.msg2$ is a valid response consistent with the specifications of pr.
   P can then continue with the execution of the protocol.
2. P will retrieve Q's response via a CTS entry but its contents are invalid. P can provably incriminate Q with the evidence it possesses:
   – Q's signed protocol commitment message;

[1] Normally these entries are published by the CTS immediately after they are received. A possible *delayed-publication* variant may enable the immediate publication of a hashed receipt only, with the publication of the full message at a later point in time, specified as part of the submitted message.

- CTS's signed entry $s_{id.cts}$ together with the contents pr and $protocol.msg1$ of the internal encrypted component;
- CTS's signed entry $s'_{id.cts}$ together with the contents pr and $protocol.msg2$ of the internal encrypted component;

3. By $ts_{cts} + \delta$ (according to CTS's clock) no response has been received with index l and id s'_{id}.
   P can provably incriminate Q with the evidence it possesses:
   - Q's signed protocol commitment message;
   - CTS's signed entry $s_{id.cts}$ together with the contents pr and $protocol.msg1$ of the internal encrypted component;

   The fact that no response entry with the specified index has been posted by the specified time can be deduced by any party by examining CTS's posted entries, or it can be also stated explicitly in a time-stamped signed statement obtained from CTS upon request [2].

In general, designing application protocols that use a CTS may involve some subtle issues. In particular, the protocols need to be designed so that in certain abnormal circumstances hidden information in various message components may need to be exposed, by the client or a delegate acting on her behalf, without however violating the interests of the client.

In the next section we demonstrate the use of CTS to enhance an electronic elections protocol by making it resistant to selective denial of service, while attaining strong integrity and confidentiality guarantees.

3 Enhanced Elections Protocol

In [12] we introduced a new electronic elections protocol that relies on the *delegated enforcement* principle to attain attractive properties. The protocol relies on two election servers A and C to attain the desirable strong integrity and confidentiality guarantees required in elections (e.g. [3,5,8]), while reducing to a minimum the amount of involvement required by each voter V_i.

Below we present an enhancement of that protocol which relies on a CTS to prevent *selective denial of service* [3]. Without such an enhancement, server A can selectively discriminate against certain groups of voters (with certain expected voting patterns), by delaying their voting session completion, and thus increasing the probability that they will terminate prematurely and so that their votes will not be recorded.

The design of such an enhanced CTS-based protocol involves some subtle issues in balancing the need to enforce the correct behavior of each participant, with the need to maintain the strict confidentiality of each voter's vote.

[2] There is actually a 4th possible scenario in which Q has responded correctly but P attempts to falsely incriminate him. We ignore this possibility here because with proper protocol design P will not have sufficient evidence to *provably* incriminate Q.

[3] Incidentally, the use of a CTS may also significantly reduce the likelihood of conventional denial of service - it is easier to widely-replicate the generic CTS service than it is to widely-replicate the more complex and specialized services provided by A or C.

3.1 *CTS*-Based Elections Protocol Steps

Step 0: Before the voting period begins, A and C publish their signed protocol commitment statements containing the protocol identifier pr_{eln} and the CTS posting-location index l_{eln}, among other details such as elections start and end times.

Step 1: After the voting period begins, each voter V_i constructs its vote submission in the following way:

$$M_{1i} = \mathcal{BL}_{r_{1i}}(ballot_i)$$

where r_{1i} is a secret random V_i generates, $ballot_i$ is the actual vote together with a random secret identifier b_i generated by V_i , and $\mathcal{BL}_{r_{1i}}(ballot_i)$ is conventional *blinding* [4] of the ballot with the random value r_{1i}.

$$M_{3i} = \{\ r_{1i}\ \}_{K_{C+}}$$

$$\mathcal{M}_i = \{\ pr_{eln}\ ,\ \{\ f_{1i}\ ,\ M_{1i}\ \}_{K_{A+}}\ ,\ \{\ f_{3i}\ ,\ M_{3i}\ \}_{K_{A+}}\ \}_{K_{V_i-}}$$

where f_{1i} and f_{3i} are secret values used as confounders [6]

$$\mathcal{X}_i = \{\ f_{2i}\ , V_i\ ,\ \mathcal{M}_i\ \}_{K_{A+}}$$

where f_{2i} is another secret confounder.

V_i submits to CTS:
$V_i\ \rightarrow\ CTS$**:** $\{\ l_{eln}\ ,\ s_{id}\ ,\ [\mathcal{X}_i\ ,\ M_{3i}]\ \}_{K_{CTS+}}$

Step 2: CTS sign and posts at l_{eln}:

$$\{\ l_{eln}\ ,\ s_{id}\ , ts_i\ ,\ s_{id.cts}\ ,\ [\mathcal{X}_i\ ,\ M_{3i}]\ \}_{K_{CTS-}}$$

Step 3: Let L_{sub} denote the CTS's signed list of entries submitted for l_{eln} and received before $ts_{end.of.voting}$:

$\mathcal{X}_j\ ,\ M_{3j}$
$\mathcal{X}_k\ ,\ M_{3k}$
. . .
. . .
. . .

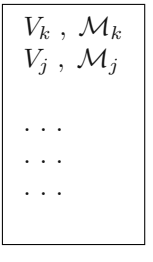
After $ts_{end.of.voting}$, A can make one of the following claims for each entry e in L_{sub}:

1. **Non-Eligible Voter** - A proves by providing the contents of $\mathcal{X}_e$: f_{2e} , V_e , $\mathcal{M}_e$
2. **Inconsistency within the First Component of an Entry** - A proves by providing the (first layer) contents of $\mathcal{X}_e$
3. **Duplicate Identical Entries** - There are two (or more) entries e , e' in which $\mathcal{X}_e = \mathcal{X}_{e'}$ and $M_{3e} = M_{3e'}$ - evident
4. **Duplicate Different Entries by Same Voter:** A proves by providing the (first layer) contents of two (or more) entries $\mathcal{X}_e$ and $\mathcal{X}_{e'}$ in which for the same V_e there are different $\mathcal{M}_e$ and $\mathcal{M}_{e'}$
5. **Duplicate-First-Component Entries:** There are two (or more) entries e , e' in which $\mathcal{X}_e = \mathcal{X}_{e'}$ and $M_{3e} \neq M_{3e'}$ - evident
6. **Inconsistency between First and Second Components of an Entry:** A proves by providing the (first layer) contents of $\mathcal{X}_e$: f_{2e} , V_e , $\mathcal{M}_e$, as well as providing the (second layer) contents f_{3e} and M'_{3e} of $\{ f_{3e} , M'_{3e} \}_{K_{A+}}$, where $M'_{3e} \neq M_{3e}$
7. **Valid Entry:** All other cases

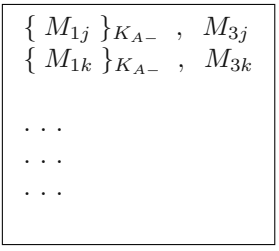
A considers as Accepted Entries:

- All Valid Entries (category 7)
- One entry for each group of Duplicate Identical Entries (category 3) (unless their contents is inconsistent in which case A opens up to prove)
- One entry for each group of Duplicate First Component Entries (category 5) (unless all have inconsistent contents in which case A opens them all up to prove)

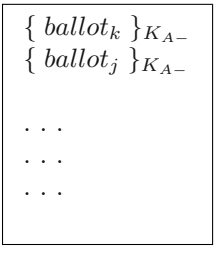
After $ts_{end.of.voting}$ A marks each entry in L_{sub} as either accepted or not accepted, according to the above rules. A generates a randomly re-ordered list L_{acc} with one entry for each accepted entry, signs the list, and posts for C at CTS:



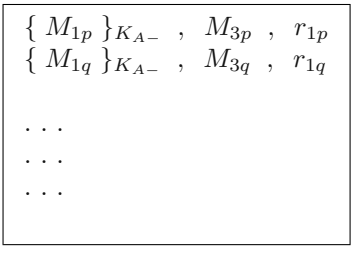
For every entry in L_{sub} which is not accepted, A posts the proof as described above (except for category 5 entries in which one of the Duplicate First Component is declared as accepted - in such a case the rest of the not-accepted entries in that group are not opened at this phase yet).
A generates a re-ordered list $L_{bl.votes}$ with one entry for each accepted entry, signs the list, and posts for C at CTS:



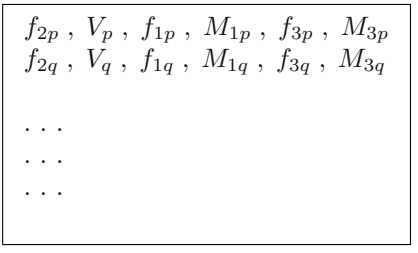
Step 4: C checks for the consistency of A's CTS postings, and posts a violation proof in case it discovers an inconsistency in A's postings. C then generates a re-ordered list $L_{final.votes}$ with one entry for each entry in $L_{bl.votes}$ for which the unblinding produced a readable ballot, signs the list, and posts at CTS:



C generates a list $L_{unreadable}$ with one entry for each entry in $L_{bl.votes}$ for which the unblinding produced an unreadable ballot, signs the list, and posts at CTS:



Step 5: A checks for the consistency of C's CTS postings and posts a violation proof in case it detects any inconsistency, provably-incriminating C. A then generates a list $L_{unreadable.proof}$ with one entry for each entry in $L_{unreadable}$, and multiple entries corresponding to multiple unopened not-accepted entries for each entry in $L_{unreadable}$ which corresponds to a Duplicate-First-Component declaration (step 3). A signs the list, and posts at CTS:



Step 6: C checks for the consistency of A's CTS postings, and posts a violation proof in case it detects an inconsistency, provably-incriminating A.

The formal correctness claim and proof are omitted from this extended abstract. Informally we state that strong integrity and confidentiality guarantees are attained:

Unless A or C is provably-incriminated,

1. *A vote will be included in the final-votes list if and only if a valid corresponding CTS initial submission was made, and*
2. *No party may unilaterally deduce who any valid voter voted for.*

In particular, neither A or C have an opportunity to discriminate against any voter (with any demographic characteristic) without being provably incriminated.

Furthermore, the protocol is designed to enable A and C to verify the consistency of each other's election processing by monitoring the public information posted at the CTS, while at the same time preventing each of them from unilaterally being able to deduce any information about the vote of any of the voters. It even prevents A and C from unilaterally deducing the votes of disqualified voters (whose submission were disqualified either because they were ineligible to vote, or because of certain mistake in their submission).

4 Conclusions

Selective denial of service may be a significant problem, in particular in cases in which a party is required to accept liability, or may otherwise have an incentive to gain advantage from maliciously performing a violation.

A *Certified-Transfer Service* may assist reduce such risks. It is designed as a scalable highly availability and highly robust service, which does not obtain any application information about the messages it processes, or even their associated application protocols, and so has no incentive to perform any violation or discrimination. The public posting by the CTS of any message it receives enables any interested party, or delegate, to monitor the correct execution of others.

Protocols that rely on a CTS will need to be designed so that proof for incriminating any parties may be obtained based on the public information at the

CTS, while at the same time always maintaining all the information secrecy requirements. The new *CTS*-based elections protocol presented here demonstrates some subtle issues involved in the design of such protocols.

References

1. R. J. Anderson, "The Eternity Service", *Proceedings of Pragocrypt '96*, 1996.
2. N. Asokan, V. Shoup, and M. Waidner "Optimistic Fair Exchange of Digital Signatures" *Proceeding of EUROCRYPT'98*, 1998, pp. 591–606.
3. J. Benaloh and M. Yung "Distributing the Power of a Government to Enhance the Privacy of Voters", *Proceedings of the 5th ACM Symposium on the Principles of Distributed Computing*, 1986, pp. 52–62.
4. D. Chaum, "Security without Identification: Transaction Systems to Make Big Brother Obsolete", *Communication of the ACM*, Vol 28, No 10, pp. 1030–1044 Oct 1985.
5. A. Fujioka, T. Okamoto, and K. Ohta, "A Practical Secret Voting Scheme for Large Scale Applications", *Advances in Cryptology – AUSCRYPT '92, Lecture Notes in Computer Science*, Vol 718, pp. 244–251, Springer-Verlag, 1992.
6. L. Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer, " Protecting poorly chosen secrets from guessing attacks" *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, June 1993.
7. Y. Lindell, A. Lysyanskaya, T. Rabin, "On the Composition of Authenticated Byzantine Agreement", Proceedings of STOC'02, Montreal, Canada, May 2002
8. R. Mercuri, "The Electronic Voting Enigma: Hard Problems in Computer Science" Talk at The Cryptography and Information Security Group of MIT's Laboratory for Computer Science, October 19, 2001.
9. S. Micali, "Certified E-mail with Invisible Post Offices" Talk at RSA Security Conference, 1997.
10. B. Salanie, "The Economics of Contracts", *MIT Press*, 1997.
11. B. Schneier, "Applied Cryptography", 2nd Edition *John Wiley & Sons*, 1996.
12. R. Yahalom "Delegated Enforcement in Secure Protocols", *2nd Workshop on Issues in the Theory of Security*, Portland, Oregon, Jan 2002, *http://www.dsi.unive.it/IFIPWG1_7/WITS2002/prog/annotated_program.html*

Protocols with Certified-Transfer Servers (Transcript of Discussion)

Raphael Yahalom

Onaro Research

Wenbo Mao: Why do you call this denial of service? This seems to me to be a subversion of this open C, which is much more serious than denial of service, or selective denial of service. It's not denial of service, it's some total attack to demolish this open C.

Reply: I refer to it as selective denial of service to emphasise that this is different to conventional denial of service, because A is using the identity of I, which it needs to have to establish that I is a valid voter.

One thing that is different about this technique, compared to conventional denial of service, is the problems that protocol participants themselves can inflict on each other. Here the server A inflicts problems on I, where both of them are eligible participants in that protocol.

I would like to point out, in relation Wenbo's comment, that one of the advantages of the certified-transfer server approach is that it actually helps with respect to conventional denial of service as well. Services such as that one are designed to be generic, highly available and simple. It is much easier to replicate that kind of service than it is to replicate either A or C which are more specialised and more complex. So if you wanted to deal with denial of service for an electronic election, which of course is one of the big problems of electronic elections, then the conventional approach would be to say "let's replicate the election service itself." But that would involve a much more complex and specialised type of operation because of the more complex nature of A and C. However if we just did replication of something generic and well-designed, then we could, by the fact that we're actually recording these submissions within that certified transferred service, significantly improve our resistance to conventional denial of service as well.

Ernie Cohen: In this case, if you trusted that at least one of A and C is honest, would it work to simply send the thing first to the one who isn't checking the credentials, have him record the sequence of votes, and then send it to the other one? It could save actually having to produce the receipts.

Reply: This is a good point. I have discussed how you need to change the protocol design to be able to allow this A and C to still monitor each other whilst still protecting the confidentiality of the vote. Your suggestion of doing it through C would work, but similar considerations would apply. Essentially you need the same sort of message formats so that you can do it with C in this particular example.

Alec Yasinac: In the situation where the intermediate results have an impact on the overall outcome, you don't have to completely keep the vote from being counted to have an influence on the election. For example, when polls come

out and they claim that one particular person has won, then that has an impact on the polling across the rest of an area. And reporting some results later than other results, because they have been manipulated maliciously, can result in a change in the outcome.

Reply: I would point out that, in this protocol, this could not happen even without the CTS. This is because A does not have access to the votes and as the actual votes are transferred to C they are never published.

Alec Yasinac: If A is reporting from a specific precinct, and that precinct is traditionally one party's, then A could in fact hold all of those votes for some period of time.

Reply: I guess, coming from Florida, you've had a chance to think about these problems!

Hiroshi Yoshiura: Why does the CTS service need to be an independent agent?

Bruce Christianson: The question is: "Why aren't you worried about the CTS doing exactly the same thing as all the others?"

Reply: Well that's a good point. I would emphasise that the interesting thing about the CTS is the fact that the protocols are designed around it, so it would have no information. And so, consequently, by definition of the selective denial of service, it will have no reason to prefer one entry to another. The idea is that the CTS does not need to know where anything comes from, and in some cases you can use an anonymizer service. There are issues about what the CTS can deduce just from the network level about where information is coming from, but at the application level, in this design, the CTS does not know who the voter is. So if you want to be extra careful you can use an anonymous channel because the CTS does not do any verification whatsoever: it's a block, it stores it, and the agreement is that A has to collect all the postings before a certain time. So CTS does not have any information.

Michael Roe: In vague relation to your point, one way round some of these problems is to make it mandatory for people to vote, but then you get lots of attacks where you might selectively discourage people from voting. I'm wondering if some of the attacks you're trying to prevent here could be prevented in the electronic case by making certain behaviour mandatory.

Geraint Price: That works in this scenario but not, for example, in your example of online trading of things where you've got orders.

Bruce Christianson: The classic example is where you re-order an appliance.

Tuomas Aura: If it's mandatory to vote then that means that the election results will only be official after all the votes have been collected, and then a single voter could prevent an election from taking place.

Bruce Christianson: In Australia, the protocol is that you have to vote or pay the government a thousand dollars. But then you find that if your name is Aardvark or something like that, you're much more likely to be selected as a candidate by a political party because the candidates are listed in surname order. [Laughter]

Reply: I would just like to summarise by saying that the election protocol is an example of the subtleties which are more general. So under some kinds of selective denial of service, a server as such a certified transfer may help. The main focus here is that if we have such services satisfying the goals we are aiming at then the design of such protocols involves some subtleties related to the trade-off between correctness enforcement and the revealing of certain secrets.

Virgil Gligor: One of the things that they say in the US is, "vote early and vote often." Suppose that some of these voters of yours vote early and then decide to deny service to voters who would follow after them? In spite of the certified transfer server, they could actually send a large number of malformed packets to A, for example, to prevent A from forwarding and playing the protocol here. That doesn't seem to be solvable within the framework of your application. For that reason I would be very careful in talking about denial of service, because one of the things we know about denial of service is that it is not an end-to-end solvable problem in general. In other words, it is not a typical security problem: end-to-end arguments do not work with denial of service. So consequently, I would be very careful with emphasising even selective denial of service. I would try to come with up with a very different set of words for describing what you're solving. You are solving a problem, but it's really not quite denial of service.

Reply: Good point.

James Malcolm: Towards the end you said some additional messages had to be transferred to certify the transfer service, does that imply that it needs to be tailored to a particular application?

Reply: No, not at all. This is because, first of all, these messages between A and C at the collected phase do not need to go through the service.

James Malcolm: But, at a later point, you said that additional evidence had to be provided...

Reply: It's just a channel for them. In fact, the only motivation to delay this transfer to the certified transfer service is if one of the parties needs to provide evidence and fails to do so. If he doesn't do it through the certified transfer service, the other party has a provable evidence to say, "that party has promised to respond to that request by that time stamp, and here it is, we do not have that response, that's a proof." Otherwise, if the other party is the guilty one, the policy that that party may adopt is to ignore requests to provide evidence later on. But essentially, from the CTS perspective, these are just blocks and it's a channel.

Virgil Gligor: Are you looking at this system to work in an open environment? In other words, could anybody actually play this protocol: could I vote anywhere, or do I have to be registered in some sense with this services to be able to vote?

Reply: No, you need to be an eligible voter as determined at A. A is the administrator who determines who the eligible voters are.

Virgil Gligor: So, in fact, your system is not an open system in that sense, there's not anybody can vote who's A.

Reply: Anybody can submit a vote, but only the votes of eligible voters would be counted.

Virgil Gligor: What I'm trying to say is that, if there is an assumption of a closedness of the system – in other words, not too many people can send votes to A – then a lot of the problems that you have might solve themselves. But if the entire Internet can send votes to A then you still have a problem to deal with: denial of service.

Reply: Right.

Bruce Christianson: How easy is it to replicate the CTS?

Reply: Well, again, I've only sort of hand-waved my way through. There are certainly some system design challenges with respect to CTS. However, I would point out that CTS compared to other application services, is simpler, in the sense that it doesn't do much.

Bruce Christianson: It is also relatively stateless.

Reply: Essentially, because of its simplicity and because it's designed with one purpose in mind, we could get better results in terms of achieving a sufficient level of availability. But, again, that's something for which I have no evidence. That is certainly an area in progress and, in particular, even the design itself of the CTS presents an interesting quiz as it is different to that of some of the other applications I mentioned I had looked at.

Wenbo Mao: You mentioned that nobody can delete a message from CTS, so will the set of messages grow?

Reply: In practice, there would be something along the lines of an advertised policy of CTS stating that any message in any protocol will be kept for a certain period of time, and then moved off-line. So as long as the rules are clear for all the participants, and they make commitments based on the policy of the CTS, then I think we're fine.

Geraint Price: You might be able to make it stateless by pushing a return and handing it back out to the client. CTS signs it and hands it back, then it is up to the client, if they actually want to prove anything at the end of the day, to keep hold of that.

An Architecture for an Adaptive Intrusion-Tolerant Server*

Alfonso Valdes, Magnus Almgren, Steven Cheung, Yves Deswarte**, Bruno Dutertre, Joshua Levy, Hassen Saïdi, Victoria Stavridou, and Tomás E. Uribe

System Design Laboratory,
SRI International,
333 Ravenswood Ave., Menlo Park, CA 94025
valdes@sdl.sri.com

Abstract. We describe a general architecture for intrusion-tolerant enterprise systems and the implementation of an intrusion-tolerant Web server as a specific instance. The architecture comprises functionally redundant COTS servers running on diverse operating systems and platforms, hardened intrusion-tolerance proxies that mediate client requests and verify the behavior of servers and other proxies, and monitoring and alert management components based on the EMERALD intrusion-detection framework. Integrity and availability are maintained by dynamically adapting the system configuration in response to intrusions or other faults. The dynamic configuration specifies the servers assigned to each client request, the agreement protocol used to validate server replies, and the resources spent on monitoring and detection. Alerts trigger increasingly strict regimes to ensure continued service, with graceful degradation of performance, even if some servers or proxies are compromised or faulty. The system returns to less stringent regimes as threats diminish. Servers and proxies can be isolated, repaired, and reinserted without interrupting service.

1 Introduction

The deployment of intrusion-detection technology on mission-critical and commercial systems shows that perfect detection and immediate mitigation of attacks remain elusive goals. Even systems developed at great cost contain residual faults and vulnerabilities. In practice, emphasis must shift from unattainable "bulletproof" systems to real-world systems that can tolerate intrusions. An *intrusion-tolerant* system is capable of self-diagnosis, repair, and reconstitution, while continuing to provide service to legitimate clients (with possible degradation) in the presence of intrusions [6]. This paper describes the conceptual

* This research is sponsored by DARPA under contract number N66001-00-C-8058. The views herein are those of the authors and do not necessarily reflect the views of the supporting agency. Approved for Public Release—Distribution Unlimited.

** LAAS-CNRS, Toulouse, France

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 158–178, 2004.

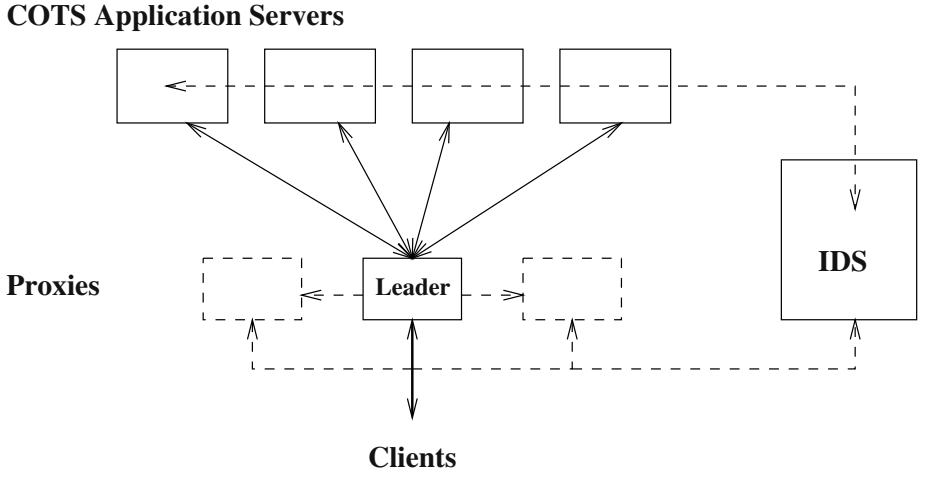


Fig. 1. Schematic view of the intrusion-tolerant server architecture

architecture of such a system, and our experience with its initial implementation. Our design integrates concepts from distributed intrusion detection, fault tolerance, and formal verification.

Our primary concerns are availability and integrity: the system must remain capable of correctly servicing requests from honest clients even if some components are compromised. The architecture is also intended to be scalable: servers and proxies can be added to improve the system's performance and reliability, depending on the resources available.

1.1 Architecture Outline

The architecture, shown in Figure 1, consists of a redundant *tolerance proxy* bank that mediates requests to a redundant *application server* bank, with the entire configuration monitored by a variety of mechanisms to ensure content integrity, including intrusion-detection systems (IDSs). The proxies and application servers are redundant in capability but diverse in implementation, so that they are unlikely to be simultaneously vulnerable to the same attack. The application servers provide the application-specific functionality using COTS products, thus reducing the cost of specialized custom-built components. The remainder of the configuration provides intrusion tolerance by detecting, masking, and recovering from attacks or nonmalicious faults.

The system is designed to provide continued reliable and correct service to external clients, even if a fraction of the application servers and proxies are faulty. The faults we consider include transient and permanent hardware faults, software bugs, and intrusions into application servers and tolerance proxies. The architecture can support a variety of client-server systems, such as database applications or Web content distribution, under the assumptions outlined below.

An important aspect of our system is a distributed management function that takes actions in response to adverse conditions and diagnoses. All platforms and network interfaces within the system are instrumented with a variety of monitors based on signature engines, probabilistic inference, and symptom detection. Given the reports from this monitoring subsystem, the management function undertakes a variety of tolerance policy responses. Responses include enforcing more stringent agreement protocols for application content but reducing system bandwidth, filtering out requests from suspicious clients, and restarting platforms or services that appear corrupt.

1.2 Assumptions

We assume that attackers do not have physical access to the configuration. We assume that no more than a critical number of servers are in an undetected compromised state at any given time.

Our agreement protocols assume that all nonfaulty and noncompromised servers give the same answer to the same request. Thus, the architecture is meant to provide content that is static from the end user's point of view. The reply to a request can be the result of substantial computation, on the condition that the same result be obtained by the different application servers. Target applications include plan, catalog, and news distribution sites.

The system content can be updated periodically, by suspending and then resuming the proxy bank, but we do not address specific mechanisms for doing so (noting only that online content updates can introduce new vulnerabilities). Survivable storage techniques [16,23] can be used in the future to build a separate subsystem to handle write operations, while retaining the current architecture for read requests.

The architecture focuses on availability and integrity, and does not address confidentiality. We do not defend against insider threat or network-flooding denial-of-service attacks.

1.3 Paper Outline

Section 2 describes the basic building blocks of our architecture. Section 3 describes the monitoring mechanisms, which aim to provide a picture of the current system state, including suspected intrusions and faults. Section 4 describes how this information is used to respond to threats, adapting the system configuration to the perceived system state. Section 5 describes the details of our implementation, and Section 6 presents our conclusions, including related and future work.

2 Architecture Components

2.1 Application Servers

In our architecture, the domain-specific functionality visible to the end user (the *client*) is provided by a number of *application servers*. These provide equivalent

services, but on diverse application software, operating systems, and platforms, so that they are unlikely to be vulnerable to common attacks and failure modes. They include IDS monitors but are otherwise ordinary platforms running diverse COTS software. In our instantiation (see Section 5), these servers provide Web content.

For our content agreement protocols to be practical, we assume there are at least three different application servers; a typical number would be five or seven. However, the enterprise can add as many of these as desired, increasing the overall performance and intrusion-tolerance capabilities.[1]

2.2 Tolerance Proxies

The central components of our architecture are one or more *tolerance proxies*. Proxies mediate client requests, monitor the state of the application servers and other proxies, and dynamically adapt the system operation according to the reports from the monitoring subsystem (described in Section 3).

One of the proxies is designated as the *leader*. It is responsible for filtering, sanitizing, and forwarding client requests to one or more application servers, implementing a *content agreement protocol* that depends on the current *regime*, while balancing the load. In the presence of perceived intrusions, increasingly stringent regimes are used to validate server replies. The regime is selected according to a chosen *policy*, depending on reports from the monitoring subsystem and on the outcome of the agreement protocol currently in use.

Optional *auxiliary proxies* monitor all communication between the proxy leader and the application servers, and are themselves monitored by the other proxies and the sensor subsystem. If a majority of proxies detect a leader failure, a protocol is executed to elect a new leader. Our current design and implementation focuses on the case of a single proxy. However, we discuss inter-proxy agreement protocols further in Section 4.4.

The tolerance proxies run only a relatively small amount of custom software, so they are much more amenable to security hardening than the more complex application servers, whose security properties are also more difficult to verify. We believe there is greater return, both in terms of security and utility, in expending effort developing a secure proxy bank than in developing hardened application servers. Since the proxies are mostly application-independent, their development cost can be amortized by re-using them in different domains.

2.3 Intrusion-Detection System

The third main component of our architecture is an *intrusion-detection system* (IDS), which analyzes network traffic and the state of the servers and proxies to report suspected intrusions. Some IDS modules execute on one or more dedicated hardware platforms, while others reside in the proxies and application servers. We describe our IDS capabilities in more detail in Sections 3.1 and 5.1.

[1] The architecture should be modified for large numbers of application servers, as discussed in Section 6.

2.4 Functional Overview

When a client request arrives, the following steps are performed:

1. The proxy leader accepts the request and checks it, filtering out malformed requests.
2. The leader forwards the request, if valid, to a number of application servers, depending on the current agreement regime.
3. The application servers process the request and return the results to the proxy leader. If sufficient agreement is reached, the proxy forwards the content to the client.
4. The regime is adjusted according to the result of the content agreement and reports from the monitoring subsystem.
5. The auxiliary proxies, if present, monitor the transaction to ensure correct proxy leader behavior.

3 Monitoring Subsystem

The *monitoring subsystem* includes a diversified set of complementary mechanisms, including the IDS. The information collected by the monitoring subsystem is aggregated into a global system view, used to adapt the system configuration to respond to suspected or detected threats and malfunctions, as described in Section 4. Diversity helps make the monitoring subsystem itself intrusion-tolerant, since it may still be effective if some of its components fail.

3.1 Intrusion Detection

Our intrusion-detection systems feature diverse event sources, inference techniques, and detection paradigms. They include EMERALD host, network, and protocol monitors [19,15,20,29], as well as embedded application monitors [1].

Different sensors cover different portions of the detection space, and have different detection rates, false alarm ratios, and operational conditions (e.g., the maximum rate of incoming events that can be handled). Their combination allows detecting more known attacks, as well as anomalies arising from unknown ones. The advantages of heterogeneous sensors come at the cost of an increased number of alerts. To effectively manage them, they must be aggregated and correlated [30,18]. Alert correlation can also detect attacks consisting of multiple steps.

3.2 Content Agreement

The proxy leader compares query results from different application servers, according to the current agreement regime, as described in Section 4.1. If two or more results fail to match, this is viewed as a suspicious event, and suspect servers are reported.

3.3 Challenge-Response Protocol

Each proxy periodically launches a *challenge-response protocol* to check the servers and other proxies. This protocol serves two main purposes:

- It provides a heartbeat that checks the liveness of the servers and other proxies. If a proxy does not receive a response within some delay after emitting a challenge, it raises an alarm.
- The protocol checks the integrity of files and directories located on remote servers and proxies.

The integrity of application servers is also verified indirectly by content agreement, as mentioned above. However, a resolute attacker could take control of several servers and modify only rarely used files. The challenge-response protocol counters this by reducing file error latency, detecting file modifications before they return incorrect content to the users.

As an integrity check, the challenge-response protocol is similar to Tripwire™ [28]. For each data item whose integrity is to be checked, a checksum is computed from a secret value (the challenge) and the content of the item. To resist possible guesses by an attacker, the checksum is computed by applying a one-way hash function to the concatenation of the challenge and the content to be checked. The resulting checksum is then compared with a precomputed one. This is sufficient to check if static data items have been modified.

However, we must also consider the possibility that an attacker with complete control of a server or proxy modifies the response program to return a correct challenge response for a file incorrectly modified. In particular, if the challenge is always the same, it is easy for the attacker to precompute all responses, store the results in a hidden part of the memory or disk, and then modify the data. The attacker would then be able to return correct responses for incorrect files.

To guarantee the freshness of the response computation, a different challenge is sent with each request. The proxy could check that each response corresponds to the specified challenge by keeping a local copy of all sensitive files and running the same computation as the server, but this imposes an extra administrative and computational load on the proxy.

Instead, we can exploit the fact that servers and proxies are periodically rebooted (e.g., once a day), as a measure for software rejuvenation [11]. In this case, a finite number C of challenges can be generated offline for each file to be checked, and the corresponding responses computed offline, too. The results are then stored on the proxy. At each integrity-check period, the proxy sends one of the C challenges and compares the response with the precomputed result. The challenges are not repeated, provided

$$C > (\text{time between reboots}) \times (\text{frequency of challenge-response protocol}) \ .$$

An attacker could circumvent this mechanism by keeping copies of both the original and the modified files. But such a large-footprint attack should be detected, either by checking the integrity of the concerned directories, or by the host monitor IDS.

3.4 Online Verifiers

As part of the design process, we express the high-level behavior of the proxy as a *reactive system* that can be formally verified. An abstraction of the system is described using a finite-state omega-automaton and the properties of interest are specified in temporal logic. The high-level specifications can be formally verified using model checking [10]. However, this does not guarantee that the implementation (the *concrete* system) meets the corresponding requirements.

To fill this gap, we introduce *online verifiers*, which check that the abstract properties hold while the concrete system is running, by matching concrete and abstract states. If an unexpected state is reached, an alarm is raised. Only *temporal safety* properties can be checked in this way [27]; however, the challenge response heartbeat described in Section 3.3 provides a complementary liveness check.

The online verifiers are generated by annotating the proxy C program source. Since type-safeness is not guaranteed, and only high-level properties are checked, this does not detect lower-level faults, such as buffer overflows. We will apply complementary mechanisms at this level, such as compilation using the StackGuard tool [2].

4 Adaptive Response

We now describe how the system responds to state changes reported by the monitoring subsystem described in the previous section.

4.1 Agreement Regimes and Policies

A main role played by the proxy leader is to manage the redundant application servers. The proxy decides which application servers should be used to answer each query, and compares the results. The number of servers used trades off system performance against confidence in the integrity of the results.

Figure 2 presents the main steps in the *content agreement protocol* executed by the proxy leader. This protocol is parameterized by an *agreement regime*, which must specify, at each point in time:

1. Which application servers to forward the request to
2. What constitutes sufficient agreement among the replies
3. Which servers, if any, to report as suspicious to the monitoring subsystem

A good agreement regime should perform load balancing and choose the application servers as randomly as possible, so attackers cannot always predict which server will service a particular request.

Most important, the regime should be dynamically adjusted in response to alerts: a *policy* specifies the action to take next if no agreement is achieved, and which regime to use in response to various events. A policy must also specify how to respond if intrusions or other adverse conditions are detected, and when to

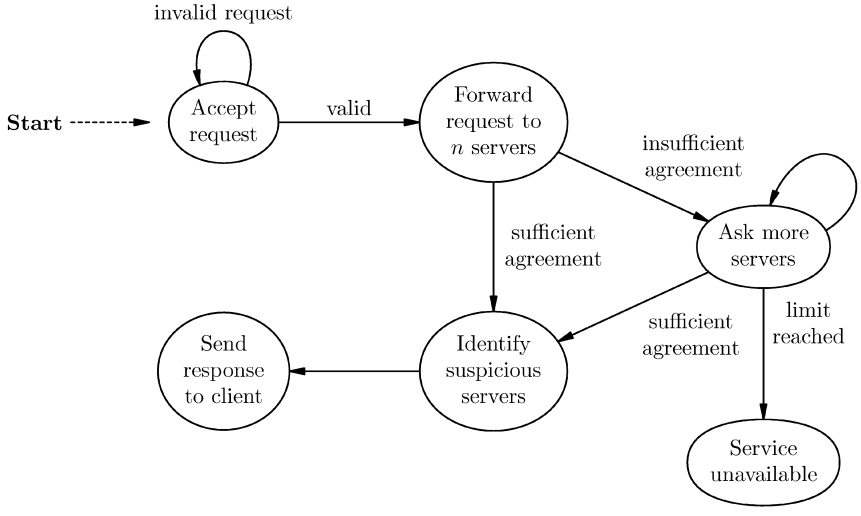


Fig. 2. Generic content agreement protocol

return to a less stringent agreement regime. The transition to a stricter regime can also occur as a result of administrative action (e.g., if the administrator is alerted by external sources).

Example 1 (Simple agreement regime). Given an architecture with N application servers, the *simple agreement regime* n, for $1 \leq n \leq N$, specifies that each client request be forwarded to n different application servers, randomly chosen, that are not considered compromised. Sufficient agreement is reached if a majority of servers (at least $\lfloor n/2 \rfloor + 1$) agree; machines in the minority are reported as suspicious. The absence of a reply is counted as a vote in the minority.

We refer to simple agreement regimes 1, 2 and 3 as *single*, *duplex* and *triplex modes*, respectively. The agreement regime where each client request is sent to all application servers is referred to as *full mode*. More complex regimes are possible:

Example 2 (Probabilistic agreement regime). A *probabilistic agreement regime* assigns to each server a real-valued *confidence*, as an estimated probability of noncompromise. Given a request, the leader chooses a set of servers such that the sum of their confidences is greater than a chosen lower bound. Results are ranked by the sum of the confidences of the machines that produced them. If the top-ranked result's confidence exceeds a given threshold, it is used. The confidence for servers in the weighted minority is reduced, and the monitoring subsystem is notified.

Table 1. Probability of sending corrupted content, querying one more server in case of a tie

F	initial simple regime n				
	1	2	3	4	5
0	0	0	0	0	0
1	0.2	0	0	0	0
2	0.4	0.3	0.3	0	0
3	0.6	0.7	0.7	1	1
4	0.8	1	1	1	1
5	1	1	1	1	1

One possible policy uses the simple agreement regimes and queries one more server if no majority is obtained; a more conservative policy could move directly to full mode instead. In the probabilistic case, more servers can be queried if the top-ranked reply does not reach the desired confidence level. Suspicious application servers are taken offline for further diagnosis and repair (see below).

Note that the agreement protocol is just one mechanism of many to guard against corruption of content. To cause the system to provide invalid content, an attacker must corrupt multiple servers simultaneously so that they provide the same incorrect content, while evading the IDS and challenge-response protocol. Any content corruption detected by these stronger mechanisms is immediately repaired. The agreement protocol is intended to make delivery of corrupt content very unlikely, as long as assumptions about the maximal number of simultaneously compromised servers hold.

4.2 Risk vs. Performance

The choice of policy poses a tradeoff between what is considered the acceptable risk of bad content delivery versus the system performance requirements. The reliability of any particular regime depends on the quality of the IDS alerts and on the fault assumptions made.

Given assumptions about the number of failed servers, we can estimate the probability that the content provided to a client under a given policy is not correct. As an illustration, consider the simple agreement regime of Example 1. Let N be the number of application servers, F be the total number of compromised servers, and n be the starting simple regime (number of queried servers). The following focuses on integrity: to cause delivery of corrupt content, an attacker must modify content identically on all compromised servers. This is the worst-case assumption from the integrity point of view.[2]

Table 1 shows the probability of sending an incorrect response for $N = 5$, using a policy that starts with the simple regime n and queries one extra server in case of a tie, assuming that all F compromised servers agree in their incorrect response. The higher regimes reduce the probability of incorrect response and increase the probability of detecting compromised servers, provided only

[2] If the attacker wishes to deny availability, content corruption can be arbitrary; no reply will be sent if the protocol fails.

a minority of servers are faulty ($F \leq 2$). In duplex mode, service integrity is guaranteed if at most one server is compromised, but some requests may receive no response, unless the policy increases the regime. In triplex mode, continued and correct service is guaranteed if one server is compromised, but the system offers a reduced capacity to clients. The full regime provides correct service (with performance degradation) if several, but not a majority, of the servers are compromised (up to 2 for $N = 5$).

4.3 Responses to Alerts

Alerts are indications of possible attacks or compromises. Experience shows that all systems connected to the Internet are regularly attacked, or at least probed. IDS components generate a large number of valid alerts that do not always indicate system compromise, as well as numerous false alarms. Other parts of the monitoring subsystem generate alerts as well: the content agreement and challenge-response protocols provide alerts that identify likely compromises.

Although alerts can arise from direct detection of an attack, many report symptoms of a compromise that has already occurred. While attack detection usually indicates only the possibility of a compromise, symptom detection can reliably recognize a compromise after it has occurred. For example, unexpected server behavior such as an outbound connection unambiguously indicates some compromise.

The system can invoke a variety of actions in response to alerts, including:

- Temporarily blocking the address from which an attack appears to originate
- Increasing the agreement regime
- Increasing the coverage and frequency of the challenge-response protocol
- Disconnecting and rebooting a server
- Refusing service and alerting the system administrator

The choice of action depends on the current system state and the nature of the alert. Alerts can be evaluated according to the *severity* of the compromise they indicate, the *detail* they provide, and the *reliability* of the alert itself. We consider three classes of alerts: warnings of probes and clearly unsuccessful attacks (detailed, reliable, but not severe), credible indications that a specific asset is compromised (detailed, reliable, and severe), and evidence of significant attacks with unknown outcome (not detailed, somewhat reliable, and potentially severe).

The response to probes and unsuccessful attacks is to temporarily block the apparent source address. We are careful not to block for too long, since this can lead to denial of service if an attacker launches a nuisance probe with a spoofed source address.

If an application server is detected as compromised or otherwise failed, the system administrator is alerted, and the server is rebooted from a read-only copy of its operating system and data. The proxies detect that the server is back online by running a version of the challenge-response protocol. The proxy recovery protocol is similar, with the difference that the operational data for the

proxy (e.g., current policy and regime, configuration) must be recovered from the other operational tolerance proxies, if present.

Adaptation is used to respond to the third class of alerts. The response to evidence of attacks with unknown outcome is to move to a more stringent agreement regime, which provides greater integrity assurance and helps identify compromised servers. The *alert manager* is a proxy module that accepts alerts from the monitoring subsystem. It assesses the state of each application server as up, down, or suspicious. This information is used to choose the appropriate agreement regime, as part of the implemented policy.

Finally, some alerts are both reliable and very severe, and indicate the compromise of a significant number of components. Such alerts arise, for example, if there is still insufficient agreement after all application servers are queried. In such cases, the only safe option is to shut down the system and alert the system administrator.

As an additional security mechanism, servers and proxies are regularly rebooted to prevent software aging and defeat incremental intrusions, as mentioned in Section 3.3. This does not fix any flaws present in the system, but gives an attacker only a limited window of time to exploit them.

4.4 Multiproxy Protocols

Our implementation to date has focused on detection and response protocols for multiple application servers mediated by a single leader proxy, and no auxiliary ones. However, we are developing the general case, where redundant proxies mitigate the weakness presented by the leader as a single point of failure (albeit a hardened and closely monitored one).

In a multiproxy configuration, auxiliary proxies monitor requests and replies from the application servers, and can thus quickly observe if the leader has failed. Like the leader, each auxiliary proxy includes an alert manager, which receives reports from the monitoring subsystem. A detected failure anywhere in the system is reported to all proxies, which must agree on the response. In particular, the proxies must maintain a consistent view of the current regime and of which proxy is the current leader.

Our multiproxy design includes three proxies and is intended to tolerate the compromise or failure of one of them. Proxies communicate with each other via a multicast channel implemented using a local ethernet link (Figure 1). This channel is assumed to ensure with very high probability that any message from one proxy is delivered to all other proxies, and that all messages are received in the same order by all proxies. In particular, this guarantees that *asymmetric* failures (a faulty proxy sending inconsistent messages to its peers) do not occur. Under this assumption, simple majority voting protocols are sufficient for maintaining consistency. (If needed, reliable atomic multicast protocols, such as [25], can be implemented to satisfy this assumption.)

When a proxy decides that a new agreement regime should be enforced, it multicasts a request to the other proxies, which reply by multicasting their

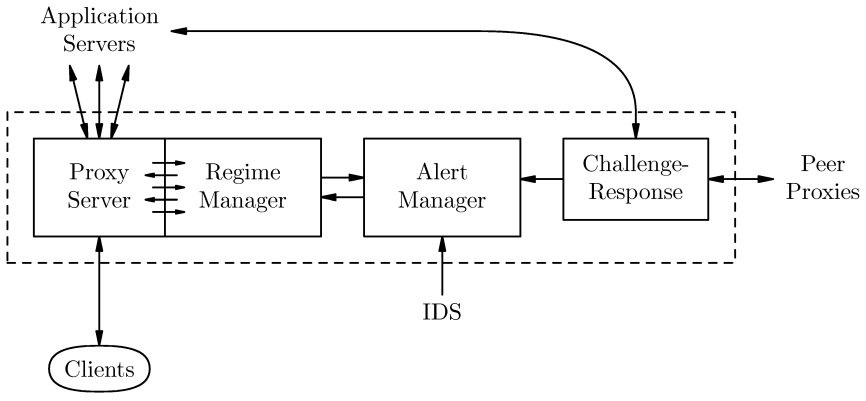


Fig. 3. High-level view of proxy implementation

agreement or disagreement. The regime is changed only if at least two of the three proxies agree. The disagreeing proxy, if any, is reported to the alert managers.

If an auxiliary proxy suspects that the leader is faulty, it multicasts a request for changing leader. If the other auxiliary proxy agrees, the leader is disconnected for repair and the administrator is alerted. The proxy that initiated the change of leader remains auxiliary and the other becomes the new leader. If the auxiliary proxies disagree on the leader status, the "accuser" is considered suspect by the other two.

A similar protocol is used if an auxiliary proxy is suspected of compromise. The decision to expel a proxy (leader or auxiliary) requires unanimity between the two others. If they do not agree, the accuser is suspect but not immediately shut down. After a delay, the accuser may persist and reinitiate the expel protocol. After a fixed number of "false accusations", the accuser is itself considered faulty and restarted. This reduces the risk of prematurely removing a noncompromised proxy that accused another by mistake.

5 Implementation

Our instantiation of the architecture provides intrusion-tolerant Web services. The first prototype has been completed, focusing on the content-agreement protocols, Web server deployment, and IDS. The Web servers used are Apache 1.3.20 running under Solaris 8 and FreeBSD 4.2, Microsoft IIS 5.0 running under MS Windows 2000, and Netscape FastTrack 4.1 (iPlanet) under RedHat 7.1.

Figure 3 shows the main components of our proxy implementation. The regime manager is responsible for executing the content agreement protocol (Figure 2). The alert manager takes input from the IDS subsystem and the challenge-reponse protocols, and notifies the regime manager when changes are warranted.

5.1 Monitoring Subsystem

The implemented monitoring subsystem includes a variety of intrusion detection sensors and alert correlation engines, as follows:

- Network-based sensors detect a variety of network attacks and probes in real time. These sensors run on a dedicated machine that monitors the traffic between the clients and the proxy, and the private subnet between the proxy and the application server bank.
  eXpert-Net is a suite of network-based sensors, each of which focuses on a specific network protocol. It features an attack knowledge base and inference engine developed using a forward-chaining rule-based system generator, P-BEST [14]. eXpert-Net can detect complex attacks and variations of known attacks. For example, by performing session and transaction reconstruction, eXpert-HTTP, a sensor for monitoring Web traffic, detects attacks that would be missed if the analysis were performed on a per-packet basis.
  Snort [26] is an open-source intrusion-detection sensor that has an extensive and updated knowledge base of attack signatures. Although both Snort and eXpert-Net are signature-based sensors, the diversity of their knowledge bases and implementation enables them to complement each other.
  eBayes-TCP [29] uses a probabilistic model-based technique. This combines the generalization potential of statistical anomaly detection with the attack specificity of signature-based detection.
- Host-based operating-system-level sensors complement the network-based sensors by monitoring events at the operating system level. EMERALD eXpert-BSM [15] analyzes the audit records generated by the Sun Solaris Basic Security Module (BSM) to perform real-time signature-based detection. We are investigating deploying STAT host-based monitors [31] and Real Secure sensors [12] to monitor the other operating system platforms, including Windows. The generated alerts must be in a common format, such as IDMEF [5], accepted by the correlation engines.
- Host-based application-level sensors couple with an application to obtain high-level, semantically rich data for intrusion-detection analyses. We have developed an Apache module to collect HTTP transaction data, which is analyzed by the signature-based sensor eXpert-HTTP [1].
- A *blue sensor* [29] discovers hosts and services, and monitors their operational status, based on Bayesian inference. This sensor is coupled to the EMERALD eBayes-TCP session monitor. As a result, detection sensitivity is increased: failed accesses to invalid hosts and services are inherently more suspicious. Moreover, false alarms are reduced: accesses to unavailable services are expected to fail, so innocent traffic accessing these services will not trigger an alert.
- An EMERALD probabilistic alert correlator [30] performs data fusion from heterogeneous sensors.

5.2 Content Agreement Using MD5 Checksums

A basic function performed by the proxy is checking that the pages returned by two or more application servers match. To improve the efficiency of this process, we use MD5 checksums [24], which the servers compute for each page served. These checksums are cryptographically strong: producing a fake page that matches a given MD5 checksum is an intractable problem given the current and foreseeable state of the art.

When comparing content from several servers, only the MD5 checksums need to be retrieved from all but one, which is queried for both checksum and content. The proxy verifies that the checksums match; if so, it also verifies that the received content matches the common MD5. This has the following advantages:

- Internal network bandwidth and proxy memory requirements are reduced
- When querying multiple servers, it is more efficient to compare the checksums than the full n pages; verifying a single MD5 is relatively inexpensive (linear-time in page size)
- The leader proxy can keep a cache of checksums, to be checked when lower agreement regimes are used. If cache hits occur, the proxy can operate at a higher assurance level despite using fewer application servers for content.

The servers have all been instrumented to include the MD5 checksum header, as specified by HTTP/1.1. The Apache Web server has experimental support for MD5 checksum headers, while the feature was manually added to both iPlanet and IIS.

The MD5s could be replaced by a more general cryptographic signature, where each application server signs its pages using a unique private key, and the proxy verifies each of the signatures. Such a signature can certify that the origin of each message is the corresponding application server, preventing spoofing on the private network (which can also be prevented using dedicated hardware). It does not, however, guarantee the integrity of the content itself. For instance, if source data files are compromised by an undetected attack, the application server will correctly sign the wrong data.

Similarly, public-key infrastructure can be used so that the end-users (clients) can certify that the origin of a served page is, indeed, the proxy server. Again, this mechanism by itself cannot guarantee the integrity of the content, but only the authenticity of the source.

5.3 Policy Implementation

Our implementation supports a variety of policies based on a generalization of the simple agreement regimes of Example 1. In general, each regime is specified by a pair (n, k), where n is the number of servers to query, and k is the minimum number of servers that yield sufficient agreement in that regime. Let N be the number of application servers, and $\mathcal{P}$ the set of pairs (i, j), where $N \geq i \geq j \geq 1$.

The policy is specified by two functions, $start : \{0, 1, 2, \ldots\} \to \mathcal{P}$ and $\delta : \mathcal{P} \to \mathcal{P} \cup \{\texttt{panic}\}$. If the alert manager assesses that F servers are suspicious, then

Table 2. A summary of security mechanisms in the proposed architecture

Security mechanism	*Intended function*
IDS	– Detects attacks by monitoring network – Detects unexpected traffic on internal network – Triggers adaptation mechanisms – Alerts system operator of serious conditions
Content agreement	– Corroborates content served to client – Detects erroneous application server (AS) behavior
Adaptive agreement policies	– Given evidence of suspicious events, reduces likelihood of serving incorrect content – Avoids querying suspicious and compromised AS
Challenge-response	– Verifies integrity of application servers and proxies – Triggers adaptation mechanisms – Provides heartbeat for liveness check
Proxy hardening	– Limits proxy vulnerabilities – Restricts communication between AS and outside world – Restricts many malformed requests forwarded to AS
Online verification	– Ensures that proxy software behaves according to specification
Regular reboot of proxies and AS	– Prevents unreliability due to software "aging" – Defeats long-term, incremental intrusions
Proxy peer monitoring	– Verifies that proxies are operating properly

$start(F) = (n, k)$ specifies the initial regime. The client request is forwarded to n servers, and a response is considered correct if there is agreement between k of them; otherwise, the function δ is used to identify a new pair (n', k'), which dictates the new regime, querying $n' - n$ extra servers. This is repeated until satisfactory agreement is obtained, or the `panic` state is reached, in which case the system is considered too corrupted to function. The alert manager is notified of the content agreement results.

Implemented policies can range from efficiency-conscious ones that initially ask only a few servers and query one additional server when needed, to integrity-conscious ones that query more servers initially and immediately query all servers when in doubt.

6 Discussion and Conclusions

Our intrusion-tolerant architecture combines a variety of traditional security mechanisms, as well as concepts from fault tolerance and formal verification. Table 2 summarizes these mechanisms, and the protective functions they play. The system is adaptive, responding to alerts and intrusions, trading off perfor-

mance against confidence in the integrity of the results. Our architecture allows for a wide variety of response policies to be implemented, depending on the environmental assumptions and cost-benefit analysis made.

6.1 An Example—Code Red Scenario

The utility of the various mechanisms and levels of detection and recovery can be illustrated using the well-known Code Red worm [17] as an example.

Both the EMERALD eXpert-Net and Snort IDSs presently installed detect the Code Red attack. The proxies are notified of the detection, and the source address of the attack is temporarily blocked. Consider now an attack that, like Code Red, depends on a buffer overflow included as part of the request, but is sufficiently different to evade the diverse IDS. Because of its length and unusual syntax, the attack request is likely to be blocked by the proxy. Note that a hardened proxy, compiled with tools such as StackGuard [2], should not be vulnerable to general buffer overflow attacks.

Suppose nonetheless that the request is forwarded to the application servers. Code Red will only affect the IIS servers, which will be a minority in a diverse system implementation. These servers will be diagnosed as failed and restarted when erroneous content is fetched from them, or as a result of running the challenge-response protocol. Meanwhile, the other servers remain available to provide valid content.

Worms such as Code Red propagate by hijacking the Web server and initiating outbound connections from the server. Such connections are detected by the IDS on the private net and blocked at the proxy, and the corresponding server is diagnosed as failed.

In summary, the attack would have to bypass a number of mechanisms to be successful, including direct detection on the external network, symptom detection on the private network, failed agreement, and challenge-response protocols. The means used by the attack to infect the system, corrupt content, and propagate are effectively thwarted.

More generally, we believe that a multiplatform attack that can corrupt content on a majority of sufficiently diverse application servers using a single query or action is very unlikely. An attack that exploits simultaneously present but different vulnerabilities requires more malicious traffic, increasing the likelihood of detection by at least one component.

6.2 Dangers and Tradeoffs

A carelessly implemented and overly reactive response policy can cause the system to operate continuously at degraded capacity. Indeed, with sufficient knowledge of the detection and response mechanisms, an attacker may launch attacks to trigger responses that result in self-denial of service.

As a simple example, it is possible to craft an HTTP request that will make different server brands respond differently, as discussed in [1]. Replacing a regular

space (ASCII 20h) with the `tab` character (09h) in a `GET` request yields a valid reply from the Apache and the iPlanet servers, but an error code from the IIS Web server. In this case, there is nothing wrong with the IIS server and it should not be rebooted. Rather, the client behaves suspiciously and should be blocked.

As mentioned in Section 2.2, the proxy forwards a sanitized version of the original request. This includes changing all whitespaces to the regular space, so we avoid such attacks.

6.3 Related Work

A number of commercial products, such as Tripwire™ and Entercept™, concentrate on protecting stand-alone Web servers, with no redundancy management. Our architecture can use these techniques as additional security measures.

SITAR [32] is an intrusion-tolerant architecture for distributed services, and shares many of our goals and assumptions. It includes adaptive reconfiguration, heartbeat monitors, runtime checks, and COTS servers mediated by proxies. While we focus on the relationship between IDSs, content agreement, and adaptive policies, SITAR focuses on using fault-tolerance techniques, including Byzantine agreement protocols, to coordinate the proxies.

The ITUA project [3] relies on intrusion detection and randomized automated response. It provides middleware based on process replication and unpredictable adaptation to tolerate the faults that result from staged attacks. HACQIT [13] uses a primary/backup server architecture that does not perform content agreement, and focuses on learning by generalizing new attacks.

Adaptive Fault Tolerance: One main characteristic of our architecture is its ability to automatically adapt the server redundancy to the level of alert. Most fault-tolerant systems adapt their redundancy to component failures, rather than the level of perceived threat. When a persistent fault is diagnosed, the faulty unit is isolated, repaired and then reinserted. In most cases, this delay is made as short as possible to maintain a similar level of redundancy for the whole mission.

Architectures such as SIFT [33] and GUARDS [21], assign different levels of redundancy to concurrent tasks according to their criticality. In other fault-tolerant architectures such as Delta-4 [22] and FRIENDS [8], different redundancy levels and techniques are assigned according to fault assumptions rather than criticality. In these architectures, as well as in AQuA [4], the redundancy levels assigned to tasks can be modified by operator commands to adapt to environment changes.

In SATURNE [7], idle computers are activated to increase the redundancy of active tasks, proportionally to task criticality. Similarly, the AFT architecture [9] dynamically adapts the task redundancy according to task criticality and real-time constraints.

6.4 Future Work

We are working towards a rational tradeoff analysis, which we see as essential to the development of automated response mechanisms and their eventual

adoption. Choosing a response policy poses difficult tradeoffs between integrity, availability, latency, and assumptions about potential threats. We will study the application of Markov decision process theory to this problem. Our goal is an analysis framework that will let us evaluate different policies given a variety of assumptions about faults, attacks, and accuracy of the IDS and monitoring subsystem.

We also want to provide more guarantees concerning the ability of the system to avoid self-denial of service as a result of a low-level attack, as well as the ability to tolerate and recover from adverse conditions in a timely fashion.

Our architecture should be modified if a large number of application servers are needed. Diversity of COTS implementation is not practical in this case. Instead, the architecture should be based on clusters, each with a diverse COTS base and proxy bank, replicating the single-cluster model that we have described. A load managing component based on our hardened proxy can forward each query to one or more clusters, depending on the current load.

Acknowledgments: We thank Dan Andersson and Phil Porras for their feedback and comments, and Ayda Saidane and Vincent Nicomette for their contribution to the challenge-response protocol.

References

1. M. Almgren and U. Lindqvist. Application-integrated data collection for security monitoring. In *Recent Advances in Intrusion Detection (RAID 2001)*, volume 2212 of *LNCS*, pages 22–36. Springer-Verlag, Oct. 2001.
2. C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, Q. Zhang, and H. Hinton. StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. In *Proc. 7th USENIX Security Conference*, pages 63–78, Jan. 1998.
3. M. Cukier, J. Lyons, P. Pandey, H. V. Ramasamy, W. H. Sanders, P. Pal, F. Webber, R. Schantz, J. Loyall, R. Watro, M. Atighetchi, and J. Gossett. Intrusion tolerance approaches in ITUA. In *Fast Abstract Supplement of the 2001 Intl. Conf. on Dependable Systems and Networks*, pages B–64, B–65, July 2001.
4. M. Cukier, J. Ren, C. Sabnis, D. Henke, J. Pistole, W. H. Sanders, D. E. Bakken, M. E. Berman, D. A. Karr, and R. Schantz. AQuA: an adaptive architecture that provides dependable distributed objects. In *17th IEEE Symposium on Reliable Distributed Systems (SDRS-17)*, pages 245–253. IEEE Computer Society Press, Oct. 1998.
5. D. Curry and H. Debar. Intrusion detection message exchange format: Data model and extensible markup language (XML) document type definition, Nov. 2001. Work in progress.
6. Y. Deswarte, L. Blain, and J.-C. Fabre. Intrusion tolerance in distributed computing systems. In *Proc. Intl. Symposium on Security and Privacy*, pages 110–121. IEEE press, May 1991.
7. J.-C. Fabre, Y. Deswarte, J.-C. Laprie, and D. Powell. Saturation: Reduced idleness for improved fault-tolerance. In *18th International Symposium on Fault-Tolerant Computing (FTCS-18)*, pages 200–205. IEEE Computer Society Press, 1988.

8. J.-C. Fabre and T. Pérennou. A metaobject architecture for fault-tolerant distributed systems: The FRIENDS approach. *IEEE Transactions on Computers*, 47:78–95, Jan. 1998.
9. O. Gonzalez, H. Shrikumar, J. Stankovic, and K. Ramamritham. Adaptive fault tolerance and graceful degradation under dynamic hard real-time scheduling. In *18th IEEE Real-Time Systems Symposium (RTSS '97)*. IEEE Computer Society Press, Dec. 1997.
10. G. J. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall, Engelwood Cliffs, NJ, 1991.
11. Y. Huang, C. Kintala, N. Kolettis, and N. Fulton. Software rejuvenation: Analysis, module and applications. In *25th Symposium on Fault Tolerant Computing*, pages 381–390. IEEE Computer Society Press, June 1995.
12. Real Secure server sensor policy guide version 6.0, May 2001. `http://www.iss.net`.
13. J. E. Just and J. C. Reynolds. HACQIT (Hierarchical Adaptive Control of QoS for Intrusion Tolerance). In *17th Annual Computer Security Applications Conference*, 2001.
14. U. Lindqvist and P. Porras. Detecting computer and network misuse through the production-based expert system toolset (P-BEST). In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pages 146–161. IEEE press, May 1999.
15. U. Lindqvist and P. Porras. eXpert-BSM: A host-based intrusion detection solution for Sun Solaris. In *Proc. of the* 17^{th} *Annual Computer Security Applications Conference*, Dec. 2001.
16. P. Liu and S. Jajodia. Multi-phase damage confinement in database systems for intrusion tolerance. In *Proc. 14th IEEE Computer Security Foundations Workshop*, pages 191–205, June 2001.
17. R. Permeh and M. Maiffret. `.ida` "Code Red" worm. Security Advisory AL20010717, eEye Digital Security, July 2001. `http://www.eeye.com/html/Research/Advisories/AL20010717.html`.
18. P. Porras. Mission-based correlation. Personal communication, SRI International, 2001. `http://www.sdl.sri.com/projects/M-correlation`.
19. P. Porras and P. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *National Information Security Conference*, Oct. 1997.
20. P. Porras and A. Valdes. Live traffic analysis of TCP/IP gateways. In *Proc. Symposium on Network and Distributed System Security*. Internet Society, Mar. 1998.
21. D. Powell, J. Arlat, L. Beus-Dukic, A. Bondavalli, P. Coppola, A. Fantechi, E. Jenn, C. Rabéjac, and A. Wellings. GUARDS: A generic upgradable architecture for real-time dependable systems. *IEEE Transactions on Parallel and Distributed Systems*, 10:580–599, June 1999.
22. D. Powell, G. Bonn, D. Seaton, P. Veríssimo, and F. Waeselynck. The Delta-4 approach to dependability in open distributed computing systems. In *Proc. 18th Int. Symp. on Fault-Tolerant Computing Systems (FTCS-18)*, pages 246–251. IEEE Computer Society Press, June 1988.
23. G. R. Ranger, P. K. Khosla, M. Bakkaloglu, M. W. Bigrigg, G. R. Goodson, S. Oguz, V. Pandurangan, C. A. N. Soules, J. D. Strunk, and J. J. Wylie. Survivable storage systems. In *DARPA Information Survivability Conference and Exposition II*, pages 184–195. IEEE Computer Society, June 2001.
24. R. Rivest. The MD5 message digest algorithm. Internet Engineering Task Force, RFC 1321, Apr. 1992.

25. L. Rodrigues and P. Verissimo. xAMp: a multi-primitive group communications service. In *11th Symposium on Reliable Distributed Systems*, pages 112–121, Oct. 1992.
26. M. Roesch. Snort: Lightweight intrusion detection for networks. In *USENIX LISA'99*, Nov. 1999. `www.snort.org`.
27. F. B. Schneider. Enforceable security policies. *Information and System Security*, 3(1):30–50, 2000.
28. Tripwire white papers, 2001. `http://www.tripwire.com`.
29. A. Valdes and K. Skinner. Adaptive, model-based monitoring for cyber attack detection. In *Recent Advances in Intrusion Detection (RAID 2000)*, pages 80–92, Oct. 2000.
30. A. Valdes and K. Skinner. Probabilistic alert correlation. In *Recent Advances in Intrusion Detection (RAID 2001)*, volume 2212 of *LNCS*, pages 54–68. Springer-Verlag, Oct. 2001.
31. G. Vigna, S. Eckmann, and R. Kemmerer. The STAT tool suite. In *DISCEX 2000*. IEEE press, Jan. 2000.
32. F. Wang, F. Gong, C. Sargor, K. Goseva-Popstojanova, K. Trivedi, and F. Jou. SITAR: a scalable intrusion tolerance architecture for distributed server. In *Second IEEE SMC Information Assurance Workshop*, 2001.
33. J. Wensley, L. Lamport, J. Goldberg, M. Green, K. Levitt, P. Melliar-Smith, R. Shostack, and C. Weinstock. SIFT: the design and analysis of a fault-tolerant computer for aircraft control. *Proc. IEEE*, 66:1240–1255, Oct. 1978.

An Architecture for an Adaptive Intrusion-Tolerant Server (Transcript of Discussion)

Victoria Stavridou

System Design Laboratory, SRI International

Ernie Cohen: I didn't quite understand what it is that you're model checking. What is the model?

Reply: We put together an abstraction of the proxy; what it is supposed to be doing. That is specified as an automaton. Then you can do a model checking that verifies that the abstraction that you want has the right temporal/liveness/safety properties. Then you compile those properties into blocks of C code. You then take the code of the proxy with the properties that are expected to be there in the abstraction and then, as you run, you keep executing those blocks which will tell you if you're doing the right transitions or not.

Ernie Cohen: So it is like the alternative way to do it? For most policies you can just write them as a regular expression.

Reply: But remember this is not policy, these are properties the proxy is supposed to have. It's not a policy.

Ryoichi Sasaki: Your paper is very interesting. I want to know, when you talk about intrusion, whether this method can be used for this with a computer?

Reply: If you look at the paper, there is a section that describes how we defend against that. One of the obvious things to say is that you will know that something has gone wrong and block the URL of the server that will send out a request. Even though that server is infected, no bad request will ever go back to the client, therefore, although the thing itself suffers, it doesn't propagate any damage.

Roger Needham: I find it difficult to believe that you would ever be able to get reasonable probability estimates that a server has been compromised.

Reply: I agree actually, I think it's the good old finger in the wind.

Anon: It would be wise to do without fingers if you are running software from some vendors.

Bruce Christianson: Let's not get into the question of how many fingers we're holding up. The issue is that sometimes you end up with a policy that's very sensitive to those parameter values and where a small variation in probability leads to large changes. Clearly if that isn't the case then it doesn't matter so much.

Supporting Imprecise Delegation in KeyNote

Simon N. Foley

Department of Computer Science,
University College, Cork, Ireland.
s.foley@cs.ucc.ie

Abstract. With decentralized authorization comes the challenge of ensuring that when a credential is written, then it precisely captures the delegation that is intended. A request for a particular service may be unexpectedly rejected, despite the requester having credentials for what should be considered a comparable service. This paper considers how techniques that support imprecision in Case-Based Reasoning Systems might be used when delegating and checking authorizations in the KeyNote trust management system.

1 Introduction

Cryptographic delegation certificates/credentials specify delegation of authorization between public keys. When an application receives a request to execute a particular action, then authentication notwithstanding, the application uses credentials provided by the requester to determine whether or not the request is authorized. Trust Management approaches such as KeyNote [3] and SPKI [6] provide assistance to applications in making these decisions. Trust Management facilitates a decentralized approach: authorization may be determined without having to consult some central authorization server, and users may choose to further delegate their authority without having to refer to a Central Authority.

Applications that use KeyNote [3] to provide decentralized authorization are emerging [1,5,7]. For example, the Apache web server (version 1.3.6, and higher) uses KeyNote credentials to provide authorization for web-page access. As applications that make authorization decisions based on schemes such as KeyNote proliferate, then the numbers of users maintaining credentials will rise. That users may choose to further delegate their authorization will further encourage the use of trust management, , both in open and in closed systems.

With this decentralized authorization approach comes the possibility that users may be delegated authorizations that may not quite match their expectations. For example, Alice likes the particular service that Bob's mobile phone credential authorizes, she makes a duplicate, and requests Bob's service provider for (a credential authorizing) an identical service. However, suppose that Bob is a business and Alice is a private individual, then it is possible that Alice's request will fail as she does not hold business credentials. Rather than failing, Alice may be willing to accept an alternative credential offering authorization for an approximately similar service, to some degree that is acceptable to Alice.

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 179–188, 2004.

This paper explores how the KeyNote trust management system might be used in practice to support such degrees of imprecision when making authorization and/or delegation judgments. Supporting imprecision for information retrieval systems has been extensively considered in the literature on similarity-based retrieval for Case-Based Reasoning (CBR) systems [10,9]. Imprecision permits answers that may not formally meet the query condition, but can be considered 'close enough'. For example, a property database search for two bedroomed apartments with monthly rent under $1,000 might return, as one of its less-relevant answers, a three bedroomed apartment with monthly rent $900. We draw on these case based reasoning techniques to provide assistance in programming KeyNote credentials and making authorization decisions.

Section 2 briefly describes the KeyNote approach to decentralized authorization. The reader should refer to [3,4] for a more detailed exposition. Section 3 considers how similarity measures can be used to introduce controlled imprecision during authorization. The reader is referred to [10,9] for an introduction to similarity measures and Case Based Reasoning. Section 4 considers the problem of ensuring a consistent treatment of similarity during certificate reduction. Our solution to this problem also provides a direction for a practical implementation.

2 Authorization and Delegation

Credentials bind public keys to the authorization to perform various actions. For example, Alice may hold a credential, signed by her manager binding her public key to the authorization to have read access to a particular file. In practice, authorization is achieved by a collection of credentials that exhibit the necessary trust relationships between their keys. For example, we trust Alice's public key (for read) if her manager's public key is trusted to delegate read, and so forth, along a delegation chain that ends in a key that is known to be appropriately trusted. Given a policy (public keys, trusted in known ways), and a collection of credentials, then a network application must determine whether a particular public key is authorized to request a particular action. This *compliance checking* is done by the trust management system.

Example 1 A telecommunications company provides service of varying quality to private and business customers. A service provider with public key `Ksteve` is trusted to grant access authorization to new customers. This is expressed in terms of the following KeyNote credential.

```
Authorizer: "POLICY"
Licensees:  "Ksteve"
Conditions: app_domain=="telephone" &&
       ( (Cust=="private" && 0<=@Qual && @Qual<=1)
       ||(Cust=="business" && 1<=@Qual && @Qual<=2));
```

This *policy* credential defines the conditions under which requests from the licensee key `Ksteve` may be trusted by the telecommunications company. These

conditions are defined using a C-like expression syntax in terms of *action attributes* `app_domain`, `Cust` and `Qual` which characterize the circumstances of a request. In this case, the service provider has been granted authorization to establish `telephone` connections as a `private` or `business` customer, with quality in the ranges [0..1] and [1..2], respectively. Note that the KeyNote operation `@Qual` converts the string attribute variable `Qual` to the integer value it represents. For the purposes of illustration we use names to represent public keys (and do not provide signatures).

The service provider has the authority to delegate this trust to customers (represented by their public keys). Alice is granted authorization as a private customer (quality 1). She in turn, decides to permit Angela to use this service at a lower quality. This is reflected by the following credentials.

```
Authorizer: "Ksteve"
Licensees:  "Kalice"
Conditions: Cust=="private"
          && @Qual<=1;
```

```
Authorizer: "Kalice"
Licensees:  "Kangela"
Conditions: Cust=="private"
          && @Qual==0;
```

Customer Angela presents these credentials when requesting a (`app_domain`) `telephone` connection as a (`Cust`) `private` customer with (`Qual`) quality 0. This results in a compliance check by the Trust Management system which finds a valid delegation chain from the trusted key `Ksteve` to `Kangela`, and thus her request is authorized, △

3 Similarity Measures

Conventional Trust Management systems can be regarded as relying on an absolute similarity measure $\_ \approx \_$, where authorizations x and y are similar ($x \approx y$), if and only if they are equal. Generalizing this to a relative measure allows a degree of similarity to be given: $x \approx y$ returns a value indicating the degree of similarity between x and y, ranging from 0 (not similar) to 1 (identical). For example, for telephone authorization one might define that

$$[\texttt{Cust=private}, \texttt{Qual=1}] \approx [\texttt{Cust=business}, \texttt{Qual=1}] = 0.8$$

$$[\texttt{Cust=private}, \texttt{Qual=1}] \approx [\texttt{Cust=private}, \texttt{Qual=0}] = 0.9$$

This reflects that authorizations for private and business customers with service quality 1 are less 'similar' than for that for two private customers with service quality 0 and 1, respectively.

Definition 1 Given a relative similarity measure, then an extended compliance check to determine whether a key K is authorized for action x must check whether K is authorized for some action x', where $x \approx x' \geq n$, for a degree of similarity $0 \leq n \leq 1$ that is acceptable to the application security policy. □

EXAMPLE 2 Given `Cust` attribute values $c, d \in \{\texttt{private}, \texttt{business}\}$ and `@Qual` attribute values $0 \leq (q, r) \leq 4$, then a general similarity measure for Example 1 is defined as

$$(c, q) \approx (d, r) \equiv \begin{cases} 1.0 - 0.1 \times |q - r| & \text{if } (c = d) \\ 0.8 - 0.1 \times |q - r| & \text{otherwise} \end{cases}$$

As noted above, this reflects a belief that the authorizations of a business and a private customer with the same quality are 'different' by 0.2. When the quality of one customer is incremented/decremented, then the similarity decreases by 0.1, and so forth. The following table reflects the similarity measure calculated for the authorizations that can be delegated by the service provider `Ksteve`. The pairs (c, q) represent (`Cust`,`Qual`) values.

≈	(p,0)	(p,1)	(b,1)	(b,2)
(p,0)	1.0	0.9	0.7	0.6
(p,1)	0.9	1.0	0.8	0.7
(b,1)	0.7	0.8	1.0	0.9
(b,2)	0.6	0.7	0.9	1.0

Given the credentials from Example 1, then a Trust Management system that accepted a similarity degree of 0.8 or higher, should authorize Alice's (private customer, quality $\leq$ 1) request to make a business call with quality 1 since $(\texttt{p,1}) \approx (\texttt{b,1}) = 0.8$, but not a business quality call with level 2. Angela (`private` customer, quality 0), would be authorized to make a private quality 1 call, but would not be permitted to make business quality calls. The application using the trust management system can increase or decrease its authorization flexibility by adjusting the similarity degree. △

4 Similarity and Certificate Reduction

In addition to judgments on authorization, a Trust Management system may also be used to provide assistance in determining whether it is safe to write and sign a new credential [8]. A principal presents a set of credentials and asks for a new credential containing a specific authorization. If the principal is authorized then it is safe to write and sign a new single credential. This *certificate reduction* [6] can be used to reduce the size of delegation chains and may also provide for a degree of anonymity [2]: long delegation chains may reveal sensitive information about how authority was acquired. For example, if Steve reduces the certificate chain to Angela's authority (Example 1) by issuing a single credential for the authorization she holds, then Angela may use the telecommunications service without revealing how she acquired the authority.

In this paper we consider only *defacto* delegation [8], that is, delegation in terms of a certificate reduction that does not confer any additional authorization on the requester. We do not consider *dejure* delegation [8] where the requester may acquire authorization, in addition to what is held. However, we note in [8]

that both approaches may be implemented in terms of certification reduction and, therefore, the results in this paper are applicable to dejure delegation.

Delegation by reduction may generate a cascading effect along delegation chains and make the similarity approach ineffective. For example, a trust management system that accepts a similarity degree of 0.8 or higher, would not permit Angela (Examples 1,2) to receive a business service with quality 1, since $(\texttt{p,0}) \approx (\texttt{b,1}) = 0.7$. However, in this case, Angela is effectively authorized to receive a private service with quality 1. Therefore, she may present her delegation chain and ask `Ksteve` to reduce it and write the following credential for this service (similarity 0.8).

```
Authorizer: "Ksteve"
Licensees:  "Kangela"
Conditions: Cust=="private"
          && @Qual==1;
```

She may now present this new credential when requesting a business service with quality 1 (similarity 0.9), which is authorized.

This cascade problem can be prevented if credentials incorporate conditions relating the degree of similarity and/or approximation for which the credential was signed. In the situation above, Steve should record within Angela's credential the fact that it was issued based on authorization credentials presented with degree of similarity 0.8.

We use the following informal strategy when writing KeyNote credentials. Given a similarity relationship $\approx$ between the possible attribute values for an action attribute variable `Action`, then introduce an additional action attribute `Degree` and codify

$$\texttt{\&Degree <= Action} \approx X$$

within any credential that authorizes the X action to its licensee. Note that `&Degree` converts the string attribute `Degree` to a floating point number. The policy credential specifies, as policy, the degree of similarity n that is acceptable by including

$$\texttt{\&Degree >=}\ n$$

A compliance check by a trust management system to determine whether a key is authorized for action X must find a suitable value V for attribute `Degree` such that K is authorized for `Action=`X and `&Degree=`V. The value V may be regarded as reflecting how 'approximate', in this case, the authorization for A is. If a new credential is to be written for this authorization then it should reflect this degree of approximation, that is,

$$\texttt{\&Degree <= (Action} \approx X) - (1 - V)$$

For purposes of flexibility, the compliance check should find the *highest* possible value for V.

Example 3 Continuing Examples 1,2, private customer Alice, authorized for service qualities {0,1}, is issued the following revised credential.

```
Authorizer: "Ksteve"
Licensees:  "Kalice"
Conditions:
  Cust=="private" ->{&Degree <= 1.0;};
  Cust=="business"->{&Degree <= 0.9-0.2*(&Qual-1.0);};
```

The absence of an absolute value function `abs()` in KeyNote has resulted an a somewhat counter-intuitive coding of the $\approx$ relation in the credential. The reader may confirm that this correctly codifies the $\approx$ relation for authorizations that can be delegated from `Ksteve` (values from the table in Example 2).

A revised policy credential for `Ksteve` specifies that similarity to a degree of 0.8 or higher is acceptable:

```
Authorizer: "POLICY"
Licensees:  "Ksteve"
Conditions: &Degree>=0.8 &&
       app_domain=="telephone" &&
       ( (Cust=="private"  && 0<=@Qual && @Qual<=1)
       ||(Cust=="business" && 1<=@Qual && @Qual<=2));
```

Given these credentials, a compliance check verifies Alice's authorization to request a business quality 1 call, with a degree value of 0.9. That is, a compliance check, for action authorizer `Kalice` and action attribute values `Cust="private"`, `Qual="1"` and `Degree="0.9"`, returns `true`. Alice's request for a business quality 2 call would fail since no degree attribute value exists that satisfies the credential conditions along the delegation chain from `Ksteve` to `Kalice`.

Suppose that Alice issues the following credential to Angela:

```
Authorizer: "Kalice"
Licensees:  "Kangela"
Conditions: Cust=="private" &&
   &Degree <= 1.0 - 0.1*(@Qual);
```

Angela is primarily authorized for quality-0 private service. However, since the policy states that a similarity degree of 0.8 or higher is acceptable, then Angela is also authorized for quality-1 private service. In this case, the compliance check holds when the degree attribute is 0.9. If Angela had asked Steve to perform a reduction on these credentials, then the following credential would be generated.

```
Authorizer: "Ksteve"
Licensees:  "Kangela"
Conditions: Cust=="private" &&
   &Degree <= 0.9 && @Qual==1
```

Angela cannot use this credential to gain cascaded authorization for business service. △

EXAMPLE 4 A mobile phone service provider issues service credentials based on the following features.

0. `rc`: receive an incoming call,
1. `sc`: make an outgoing call,
2. `rv`: retrieve voicemail messages,
3. `rt`: receive text messages,
4. `st`: send text messages,
5. `rd`: receive data, such as fax,
6. `sd`: send data, such as fax,
7. `re`: retrieve email messages,
8. `se`: send email messages,
9. `uw`: use the web.

The feature set for a particular mobile phone is coded as a ten bit vector A that is indexed by $i \in [0..9]$, corresponding to the features $\texttt{rc}, \texttt{sc}, \ldots, \texttt{uw}$. $A_i = 1$ or $A_i = 0$ indicates the presence or absence of feature i, respectively.

One simple similarity measure between feature sets averages the number of different features, that is, $1 - (\sum_{i=0}^{n-1}(A_i \oplus B_i))/n$, where $(A_i \oplus B_i)$ gives the exclusive-or of the i^{th} bit positions of vectors A and B, and n is the total number of features. Including weights for each feature, generalizes this to the weighted nearest neighbor formula:

$$A \approx B \equiv 1 - \frac{\sum_{i=0}^{n-1} w_i \times (A_i \oplus B_i)}{\sum_{i=0}^{n-1} w_i}$$

where w_i gives the weight associated with feature i. Several Case Based Reasoning systems use a weighted nearest neighbor formula to provide the basis of a similarity measure between feature sets [10].

A challenge in using the nearest neighbor formula is the determination of suitable weights for features. For the current example, we assume that the weights are determined on the basis of the cost of the feature, and that the features above are listed in increasing order of cost. Basic voice communication is cheapest, while web browsing is most expensive.

For the sake of simplicity, suppose that $w_i = 2^i$, that is, feature costs rise exponentially with the value of i. While this is unrealistic in practice, it suits the purpose of the current example since the nearest neighbor formula above simplifies to

$$A \approx B \equiv 1 - \frac{A \oplus B}{2^n - 1}$$

when A and B are regarded as unsigned integers and $A \oplus B$ is a bitwise exclusive-or operation on these numbers.

Given the above weighting scheme, then the feature set with authorizations `rc`, `sc`, `rv`, `rt` and `st` is represented by the unsigned integer 31. This corresponds to the sum of weights associated with `rc` (2^0), `sc` (2^1), `rv` (2^2), `rt` (2^3), `st` (2^4). Using the general credential writing strategy described above, a KeyNote credential that delegates authorization for these features to Alice could be written as:

```
Comment: Authorize rc+sc+rv+rt+st (=31)
Authorizer: "Ksteve"
Licensees: "Alice"
Conditions: &Degree <= 1.0-(@Ops[+]31)/1023.0;
```

Where [+] represents an exclusive or operation[1]. Suppose that the following policy was in effect.

```
Authorizer: "POLICY"
Licensees: "Ksteve"
Condition: App_Domain == "Mobile" &&
           &Degree >= 0.91
```

Under this policy, Alice is also authorized to receive or to send data since we can compute:

$$\{\mathtt{rc}, \mathtt{sc}, \mathtt{rv}, \mathtt{rt}, \mathtt{st}\} \approx \{\mathtt{rc}, \mathtt{sc}, \mathtt{rv}, \mathtt{rt}, \mathtt{st}, \mathbf{rd}\} = 0.968$$
$$\{\mathtt{rc}, \mathtt{sc}, \mathtt{rv}, \mathtt{rt}, \mathtt{st}\} \approx \{\mathtt{rc}, \mathtt{sc}, \mathtt{rv}, \mathtt{rt}, \mathtt{st}, \mathbf{sd}\} = 0.937$$

However, Alice is not permitted to simultaneously receive *and* send data since,

$$\{\mathtt{rc}, \mathtt{sc}, \mathtt{rv}, \mathtt{rt}, \mathtt{st}\} \approx \{\mathtt{rc}, \mathtt{sc}, \mathtt{rv}, \mathtt{rt}, \mathtt{st}, \mathbf{rd}, \mathbf{sd}\} = 0.906$$

which is below the degree of similarity that is acceptable by the policy. This requires some explanation. Authorization for feature `rd` or `wd` is taken to mean that the user may open a connection to receive data or to send data, respectively. It is assumed that opening a connection to send and receive data requires a credential that authorizes both, simultaneously. The policy above ensures that Alice, while she may send and may receive data, cannot open a two-way connection, corresponding to a modem connection, for example. △

5 Discussion and Conclusion

This paper considers how similarity techniques that are used by case-based reasoning systems might be adapted to support degrees of imprecision when delegating authority based on KeyNote credentials. Imprecision based compliance checking permits an operation when the credentials provided do not strictly provide the authorization, but can be considered 'close enough'. Similarity measures are used to specify how 'close' authorizations for different actions are to each other.

Section 3 redefines compliance checking in terms of similarity measures; a key is considered authorized for an action if it is authorized for another similar action, to some degree of similarity. This may lead to inconsistencies resulting from cascading effect along delegation chains. Section 4 considers this problem and describes how, by codifying the similarity measure within the credentials, it can be avoided.

The practical approach taken in this paper is informal. We have proven elsewhere, using a formal model of delegation, that our general strategy is consistent. Given certain consistency properties on $\approx$ then the compliance check defined in

[1] This is used for demonstration purposes only. The current KeyNote specification [3] does not provide bitwise operations; their incorporation would be straightforward.

Section 3 can be re-cast in terms of a conventional compliance check by transforming credentials to extended credentials that appropriately encode the $\approx$ measure. A benefit of this result is that it provides a direction for the implementation of the compliance check. Given the action attributes of a request, then a value for `Degree` must be found such that the usual KeyNote compliance check holds. A KeyNote compliance check corresponds to a search of the delegation graph for a valid path to the action authorizer. This search can be modified to consider suitable values for `Degree`.

The approach proposed in this paper can be regarded as providing a basis for a security 'dial' for application systems. The dial controls the degree of imprecision in authorization that the system is willing to tolerate; it can be tuned to reflect the desired level of flexibility. This paper demonstrates that support for such an approach is feasible. Further research is needed that applies the experience that Knowledge Engineers have in developing similarity measures, to the development of similarity measures for authorization.

Acknowledgments. The motivation for Example 4 is due to Thomas Quillinan. My thanks to him for many thoughtful discussions on this work.

References

1. Apache-ssl release version 1.3.6/1.36. Open source software distribution. Available from URL `http://www.apache.org`.
2. T. Aura and C. Ellison. Privacy and accountability in certificate systems. Technical Report HUT-TCS-A61, Helsinki University of Technology, Laboratory for Theoretical Computer Science, 2000.
3. M Blaze et al. The keynote trust-management system version 2. September 1999. Internet Request For Comments 2704.
4. M Blaze et al. The role of trust management in distributed systems security. In *Secure Internet Programming: Issues in Distributed and Mobile Object Systems*. Springer-Verlag Lecture Notes in Computer Science, 1999.
5. M. Blaze, J. Ioannidis, and A.D. Keromytis. Trust management and network layer security protocols. In *Security Protocols International Workshop*. Springer Verlag LNCS, 1999.
6. C Ellison et al. SPKI certificate theory. September 1999. Internet Request for Comments: 2693.
7. S.N. Foley, T.B. Quillinan, and J.P. Morrison. Secure component distribution using WebCom. In *Proceeding of the 17th International Conference on Information Security (IFIP/SEC 2002)*, Cairo, Egypt, May 2002.
8. S.N. Foley. Trust management and whether to delegate. In *International Workshop on Security protocols*, Cambridge, UK, 2001. Springer.
9. H. Osborne and D. Bridge. Models of similarity for case-based reasoning. In E.Cambouropolos M.Ramscar, U.Hahn and H.Pain, editors, *Procs. of the Interdisciplinary Workshop on Similarity and Categorisation*, pages 173–179, 1997.
10. I. Watson and F. Marir. Case based reasoning review. *The Knowledge Engineering Review*, 9(4), 1994.

Supporting Imprecise Delegation in KeyNote (Transcript of Discussion)

Simon N. Foley

Department of Computer Science,
University College, Cork, Ireland

John Ioannidis: Remember that KeyNote does not return just `true` or `false`, but it can return a range of values. I wonder if you can use that range of values as part of the evaluation.

Angelos Keromytis: Since you were able to express the distance as an XOR I would be inclined to believe that you could express the similarity as a continuous spectrum of return values. Basically the toggle is where the cut off point is, between 0 and 1.

Matt Blaze: This reminds me vaguely of fuzzy arithmetic.

Reply: Yes, it is just like that. However, the whole framework is based on providing a suitable similarity relationship. Now, with a fuzzy arithmetic, you just have a fixed similarity relationship, or what would be the equivalent of a fixed fuzzy relationship. A knowledge engineer codes up the desired or assumed similarities between the different authorisations based on their understanding of the semantic domain. For example, define a similarity relation over file access whereby append access right and write access right are more similar than read access right and write access right.

Matt Blaze: So I guess this similarity is directly analogous with a certainty factor.

Reply: Yes.

Stefano Bistarelli: Did you ever look at the work of Gene Freuder about name-role interchangeability in constraints[1]? His idea is very similar.

Reply: Yes, however, I would emphasize that what we are using is an existing compliance.

Stefano Bistarelli: Yes. The work with Gene was in constraints, but with fixed constraints. He just finds an algorithm to define which elements can be changed in the solution and still give the same result, and hence similarity. That could be applied here.

Reply: Yes.

[1] "Eliminating Interchangeable Values in Constraint Satisfaction Problems", Eugene C. Freuder, AAAI 1991, pp 227–233.

Modeling Protocols for Secure Group Communications in Ad Hoc Networks

Alec Yasinsac[1] and James A. Davis[2]

[1] Computer Science Department, Florida State University, yasinsac@cs.fsu.edu
[2] Department of Electrical and Computer Engineering, Iowa State University, davis@iastate.edu

Abstract. Since its introduction as a communications medium, wireless technology has found broad application on the battlefield. Accompanying dramatic advances in wireless technology and the capabilities associated with small computing devices, the demand for advanced mechanisms to employ wireless technology in the battlefield continues to grow. We propose a model for deriving and reasoning about security protocols designed for battlefield use in ad hoc wireless networks. We contend that our model facilitates reasoning about protocols by integrating the communications and cryptographic aspects of battlefield group communication, and allows automated reasoning about resulting protocols. We illustrate our concept by introducing protocols to support special communication cases associated with the battlefield.

1 Introduction

The use of wireless networks is exploding as the limiting factors such as sufficient bandwidth, device size and weight, and power concerns are eliminated or mitigated. As a result, we are beginning to see the demand for small, highly mobile devices that utilize wireless communications to organize ad hoc networks that dynamically form, intercommunicate, and pass information to other wireless users and to wire-based networks, then dissolve. In this paper, we give a detailed description of one application of ad hoc wireless networks: battlefield communications. We provide a model that reflects the salient properties of the network and propose protocols that support this environment.

1.1 Communication on the Dynamic Battlefield[1]

The modern battlefield is highly dynamic. Units enter and leave the battlefield continuously. The dynamic battlefield demands several characteristics of communications, including:

[1] We use the land battlefield example because it represents a rigorously dynamic and unpredictable environment for communications. Consequently, we utilize the battlefield terminology throughout this paper. Clearly, this topic parallels the field commonly referred to by terms such as "ad hoc wireless networks" or "dynamic mobile networks". We believe if our technology is effective in a battlefield environment, it will easily meet less rigorous environments such as combat ship-to-ship communications and a wide variety of industrial scenarios.

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 189–202, 2004.

1. **Fast.** While some limited setup may be tolerated before action starts, the ability to communicate during combat should be immediate in its accessibility to the transmitter and its delivery to the recipient.
2. **Easy/transparent.** The transmitter must be able to communicate with minimal effort apart from their normal battlefield activities.
3. **Available.** Parties must be able to communicate whenever they need to.
4. **Authenticated.** The communication initiator must be able to absolutely identify all intended recipients.
5. **Private.** The communications passed during combat should not be divulged to anyone not intended for receipt.
6. **Integrity Protected.** Messages must be protected from modification during transmission.
7. **Acknowledged.** All parties to the communication must know what the other parties did and did not receive.

We do not contend that this list is all-inclusive. Further, we recognize the impact of interactions of these requirements and posit that these interactions create the bulk of the complexity involved in secure battlefield communication.

1.2 Group Communication on the Battlefield

Modern battlefield doctrine is based on mobility, flexibility, and rapid response to changing situations, yet also requires close coordination and mutually understood objectives among all members of the (command) group. This demands a group communication paradigm.

Group communication is sometimes thought of as broadcast technology. Broadcast and group communications are related, though not identical. Broadcast technology can provide efficient group communication, though group communication may or may not involve broadcasting messages.

The group communications paradigm is preferable over point-to-point connections in such an environment simply from the standpoint of reduced overhead. If the broadcast domains of the group members, the number of transmissions required to fully deliver a group message is minimized. If the broadcast domains are totally overlapping, group messages can be fully delivered with a single transmission.

While a broadcast medium enhances flexibility and efficiency, it also introduces security vulnerabilities. Since broadcast messages are available to anyone with a suitable receiver within the broadcast range, cryptography must be used to scramble messages for privacy and to authenticate intended recipients. For group communications, authentication and privacy are normally accomplished via security protocols and subsequent distribution of a group key.

1.3 Multicast Protocols to Support Group Membership

The reliance on group communication in tactical missions is critically important and a growing practice and research area. Multicasting is a popular mechanism for supporting group communication. In a multicast session, the sender transmits

only one copy of each message that is replicated within the network and delivered to multiple recipients. The multicast group was developed from the concept of the *host group model* [3] in which "a host group is a set of network entities sharing a common identifying multicast address, all receiving any data packets addressed to this multicast address by senders (sources) that may or may not be members of the same group and have no knowledge of the groups' membership". Efficient bandwidth sharing is paramount in low-bandwidth networks such as mobile ad hoc networks. Although the host group definition allows non-members to send to the multicast address, secure multicasting designates that all senders must be authorized, whether or not they are also members of the group.

Multicasting in a wireless ad hoc domain can be more complicated than multicasting in traditional wired networking. Multicasting in a wireless ad hoc network behaves closer to multicasting across a LAN than multicasting across point-to-point links.

For ad hoc networks, the scope of the multicasting can be divided into two major categories, namely intra-domain (within the ad hoc network) and inter-domain (within the ad hoc network and beyond) multicasting. Essentially, the intra-domain case can be though of as a subset of the inter-domain multicasting case.

When multiple ad hoc networks (wired or wireless) networks are linked together in multicast groups, the problem adds several degrees of complexity. In such an environment, the majority of the complexity for mapping to the wired/existing infrastructure lies with the ad hoc gateway. It is the responsibility of the gateway to map not only the multicasting connection itself between the ad hoc network protocol and the wired protocol (PIM, DVMRP, etc.) but the gateway must also manage the security on the wired domain as well. Whereas the ad hoc security mechanism would be known, the security mechanism for end-to-end service for the wired network may be dramatically heterogeneous in nature. Thus, the combination of these challenges (intra- and inter-domain) routing satisfying the security in multicast communication is a focus of this research.

1.4 Challenges to Managing Secure Groups

Not only are there numerous routing protocols, as seen in [14,16], but there are many issues associated with managing a group. The group management issue becomes more complicated when the communication needs to be secure [15]. Securing multicast communications involves distributing cryptographic keys to the members so that each can encrypt and decrypt messages as appropriate [8, 12]. To maintain the security of encrypted packets, these keys must be recalculated and redistributed at designated times or upon certain events, such as when a member joins or leaves the group. The group manager must be aware of membership changes their self, and must also propagate the consequences of these membership changes to the rest of the group.

Numerous criteria are used to analyze secure multicast solutions [12,4,16, 18]. These criteria are categorized into group membership management, network

resource consumption, receiver resource requirements, sender resource requirements, and dependency upon particular standards. These categories are elucidated below.

Group membership management criteria address the concerns of who is and is not part of the group, what the group looks like, and what happens if the group changes.

Network resource consumption criteria are concerned with the load on the network for various stages of the multicast communication process. When analyzing the bandwidth consumption of a solution, it is important to note how many messages must be transmitted each time a member joins or leaves and how large the control messages (those for managing the group) are in relation to the data messages. Also of importance is the volume of communication that can be effectively dealt with and whether the solution can handle bursty traffic.

Receiver and sender resource requirements consider the following: How many keys must each member or sender store and how large are these keys? What is the processing time involved for the member or sender to, respectively, read or send messages? Does the solution allow non-members to send data? How many senders are allowed? Must these senders be known in advance of group creation?

Dependence upon standards concern with whether the solution depends up a particular network protocol or network characteristics (such as stability, in order packet delivery, or reliable transmission)? Does the solution depend upon a particular application?

1.5 Key Distribution

Distributing keys in point-to-point environments is challenging. Keys must be generated and distributed in a way that guarantees that the security properties are upheld. One of the goals of this paper is describe a method and technology to generate and distribute group cryptography keys. A simple key management option for group keys is to distribute a single key to all members of the group. One drawback of this method is that, since the key authenticates the group, a separate mechanism must be employed to prevent the possibility that the enemy may have acquired the group key. If a single group key is compromised, it is very difficult to detect.

1.6 Group Key Mechanisms

Diffie-Hellman key agreement. The foundation of many party-to-party key distribution schemes is based on the Diffie-Hellman public key scheme [5]. Several extensions to this technique have been proposed for group key distribution mechanisms [17,1]. The now well-known Diffie-Hellman computation relies on parties being able to raise large numbers to large powers under a large modulus. The idea is elegant in its conception and is widely considered the origin of public key cryptography.

Its elegance notwithstanding, the essence of the Diffie-Hellman computation has been somewhat more fleeting. Formal methods for protocol evaluation that easily model the effects of encryption and decryption, either mathematically [10] or logically [2], struggle to represent Diffie-Hellman in their formalism, though a recent breakthrough by Meadows [11] shows promise in this area.

Wright-Fischer [7]. Fischer and Wright developed a series of group key distribution protocols based on card games. Their protocols have the favorable property that they are effective even against adversaries with unlimited computing power. Unfortunately, their protocols are computationally intensive and not suitable for practical implementations.

Harn-Kiesler [9]. Harn and Kiesler offered a key distribution scheme using a Diffie-Hellman type computation among hierarchically organized groups with key distribution centers at the apexes. A favorable characteristic of their scheme is that each key distribution need maintain only a single secret key that is used to generate the shared key with each subordinate node. A detractor is that they require extensive setup and coordination so are not well suited to the dynamic group membership requirements such as are required in one canonical ad hoc network environment: the modern battlefield.

2 A Model for Ad Hoc Battlefield Networks

As we have chosen the battlefield environment to illustrate the properties of our ad hoc networks, we now present a model of communications components that facilitates discussions about battlefield group communication activities. The sole participants in our model are communicating nodes, and we begin by defining the communicating nodes with four essential attributes: Identifier (ID), Most Recent Location (MRL), Transmission Range (XR), and Mobility Vector (MV). These attributes facilitate reasoning about immediate communication capabilities and allow nodes to predict connectivity in the dynamic environment.

In our model, application of the attribute functions returns the attribute value of the entity (e.g. $\mathrm{ID}(x)$ returns the identifier of the entity x). We introduce the following functions to allow reasoning about the relationships of nodes, where x and y are communicating nodes.

Distance. $\mathrm{dist}(x, y)$ returns the distance between node locations

Broadcast Domain. $\mathrm{BD}(x)$ returns the set of all nodes y where $\mathrm{dist}(x, y) < \mathrm{XR}(x)$.

Communications Reach. $\mathrm{CR}(x)$ returns the set of all nodes whose MRLs are within the XR of x and those that are within the communications reach of nodes in x's broadcast domain.

$$\mathrm{CR}(x) = \big\{\mathrm{BD}(x) \cup \big(\mathrm{CR}(y),\ \forall y \in \mathrm{BD}(x)\big)\big\} \quad .$$

Full Partners. $\mathrm{FP}(x)$ returns the set of all nodes y where $y \in \mathrm{CR}(x) \wedge x \in CR(y)$.

Transmitting Partners. $\mathrm{XP}(x)$ returns the set of all nodes y where $x \in \mathrm{CR}(y) \wedge y \notin \mathrm{CR}(x)$.

Receiving Partners. $\mathrm{RP}(x)$ returns the set of all nodes y where $y \in CR(x) \wedge x \notin CR(y)$.

2.1 Modeling Security Protocols for Ad Hoc Battlefield Networks

The essential operation performed in communications protocols is sending messages. Protocol outcome is determined by the data that the participants (and intruders) possess during and after these protocols. For our model, the functionality provided by the protocols requires a separate set of operators to those that model the physical communication medium.

We consider three set operations on messages: Computation (Mc), Possession (Mp), and Receive (Mr). Message Encryption (M_k) is reflected through subscript notation. Message possession is accomplished either by computing or receiving a message.

$$x \in \mathrm{Mp(msg)} \iff \big(x \in \{\mathrm{Mr(msg)} \cup \mathrm{Mc(msg)}\}\big) \ .$$

Computing messages is fundamental to protocols. Rules may be generated to allow a wide variety of computations. For example, nodes can compute messages that have been encrypted if they possess the ciphertext and the key.

$$x \in \{\mathrm{Mp}(\mathrm{msg}_k) \cap \mathrm{Mp}(k)\} \implies x \in \mathrm{Mc(msg)} \ .$$

We modify the standard paradigm used to describe security protocols by allowing messages to be sent from one participant to a communications group. We can define a Communications Group as the set of all nodes that own the group key. Specifically, $\mathrm{CG}(k)$ returns the set of all nodes that possess group key k. Essentially, CG reflects message possession, where the message possessed is a group key.

Passing messages is the canonical activity in protocols. In our model, all messages are broadcast and may be connected to a message group. The message:

$$\mathrm{send}(A, \mathrm{CG}(k), \mathrm{msg})$$

depicts participant Alice sending a message to the communications group that shares key k. It is not clear from this specification what nodes receive the message. One may speculate that every node that holds key k receives the message, but this ignores communications limitations. We define message transmission by combining the network operators with the protocols operators and express the set of message recipients as:

for nodes x, y: $x \in \mathrm{Mr(msg)}$ if $\big(\mathrm{send}(y, k, \mathrm{msg}) \wedge (x \in \mathrm{CG}(k))^2 \wedge (x \in \mathrm{CR}(y))\big)$.

[2] If the send was conducted in the clear, all nodes are considered to be in the communications group of the sender

Modeling Group Diffie-Hellman. We now exercise the model to derive a protocol specific to the battlefield environment, first given in [1] and based on the variation of the Diffie-Hellman group key exchange protocol. The setup requires each group member to establish a prior shared key[3] with the group leader, C in this case.

$$\begin{aligned}\text{Given: } & B \in \mathrm{CR}(A)\\ & C \subset \mathrm{CR}(B)\\ & \{A, B\} \subset \mathrm{CR}(C)\\ & \mathrm{Mp}(kac) = \{C\}\\ & \mathrm{Mp}(kac^{-1}) = \{A\}\\ & \mathrm{Mp}(kbc) = \{C\}\\ & \mathrm{Mp}(kbc^{-1}) = \{B\}.\end{aligned}$$

The protocol messages occur sequentially, beginning with a non-group leader. All messages are transmitted in the clear, so no communications group is specified in the send operation.

$$\begin{aligned}&\mathrm{send}(A, , (g^{na} \bmod p))\\ &\mathrm{send}(B, , (g^{nb} \bmod p, g^{nanb} \bmod p))\\ &\mathrm{send}(C, , (g^{nbnckac} \bmod p, g^{nanckbc} \bmod p))).\end{aligned}$$

Once the messages are sent, it is possible to determine who acquired the Diffie-Hellman key.

$$\begin{aligned}&\big(B \in \mathrm{CR}(A) \wedge C \in \mathrm{CR}(B) \wedge \{A, B\} \subset \mathrm{CR}(C)\big) \implies\\ &\quad \mathrm{Mc}(g^{nanbnc} \bmod p) = \{A, B, C\}\ .\end{aligned}$$

From this we see that if the members are in range of the group leader:

$$\mathrm{CG}(g^{nanbnc} \bmod p) = \{A, B, C\}\ .$$

3 Modeling a Special Group Case with Silent Partners

On the modern battlefield, there may be groups that include passive members that receive from, but need not or cannot transmit to the group as a matter of noise or emissions discipline. In this case, there must be a protocol that allows selected members to participate in group key refreshment. These protocols must allow silent members to update their group key without participating and yet, remain in the group. We represent the silent partner constraint in our model as follows:

$$x \in \mathrm{SP}(k) \iff \big(\mathrm{XR}(x) = 0\big) \wedge x \in \mathrm{CG}(k)\text{for the group key } k\big)\ .$$

[3] The private values are inverses mod p

A simple solution for including a silent partner in key renewal is to employ a proxy that will act for them. Candidates for the task (potential proxies) are easily identified:

$$x \in \mathrm{PP}(y,k) \iff (y \in \mathrm{SP}(k)) \wedge (y \in \mathrm{BD}(x)) \wedge (\{x,y\} \subset \mathrm{CG}(k)) \ .$$

The goal or our protocol is to generate a new key for the communication group of Alice, Bob, a Silent partner, and a proXy member:

$$\begin{aligned} \text{Given:} \quad & \{X,B,S\} \subset \mathrm{CR}(A) \\ & \{A,B,S\} \subset \mathrm{CR}(X) \\ & \{A,X,S\} \subset \mathrm{CR}(B) \\ \text{Derive:} \quad & \mathrm{CG}(k') = \{A,B,X,S\} \end{aligned}$$

As with the earlier protocol, the setup requires each group member to establish a prior shared key with the group leader, this time B.

$$\begin{aligned} \mathrm{Mp}(kba) &= \{B\} \qquad & \mathrm{Mp}(kba-1) &= \{A\} \\ \mathrm{Mp}(kbs) &= \{B\} \qquad & \mathrm{Mp}(kbs-1) &= \{S\} \\ \mathrm{Mp}(kbx) &= \{B\} \qquad & \mathrm{Mp}(kbx-1) &= \{X\} \end{aligned}$$

In addition, the silent partner must establish a prior shared Diffie-Hellman value with the proxy.

$$\mathrm{Mp}(ns) = \{X,S\}$$

The protocol proceeds sequentially, beginning with a non-group leader. The existing group key ensures authentication of the exchange.

$$\begin{aligned} &\mathrm{send}(A, \mathrm{CG}(k), (g^{na} \bmod p)) \\ &\mathrm{send}(X, \mathrm{CG}(k), (g^{nx} \bmod p,\ g^{ns} \bmod p,\ g^{nxna} \bmod p, \\ &\qquad g^{nsna} \bmod p,\ g^{nxns} \bmod p,\ g^{nsnxna} \bmod p)) \\ &\mathrm{send}(B, \mathrm{CG}(k), (g^{nbnsnxkba} \bmod p,\ g^{nbnanskbx} \bmod p,\ g^{nbnanxkbs} \bmod p)). \end{aligned}$$

Once the messages are sent, it is possible to determine who has acquired the Diffie-Hellman key.

$$\big(\{X,S,B\} \subset \mathrm{CR}(A)\big) \wedge \big(\{A,S,B\} \subset \mathrm{CR}(X)\big) \wedge \big(\{A,S,X\} \subset \mathrm{CR}(B)\big) \implies \mathrm{Mc}(g^{nanbnsnx} \bmod p) = \{A,B,S,X\} \ .$$

From this we can derive the result that:

$$\mathrm{CG}(g^{nanbnc} \bmod p) = \{A,B,S,X\} \ .$$

4 Reasoning about Subversion in Ad Hoc Networks

4.1 The Essence of Location

We use the term *subversion* to denote that an enemy has forcibly taken control of an asset, thereby diminishing the asset's capacity to carry out its tactical role. Within the context of communication equipment, there are several types of subversion, including: acquiring control from a distance (e.g., bogus control messages claiming to originate from a trusted source), inappropriate use of an asset due to subversion of the operator, and physical subversion (e.g., theft) that leads to an unexpected movement of the asset. For the purpose of this paper, we focus on the last problem of detecting unexpected movement of the asset.

Each asset in the battlefield has a role in supporting the tactical plan. Communication modalities, in so far as who an asset communicates with and for what reasons, are preplanned with rigorously defined procedures for deviating from the established plan. Movement of assets is also guided by the tactical plan. We desire to detect unexpected or unplanned movement of an asset, which may indicate subversion.

The model set forth in Section 2 allows us to reason about the movement of assets. The notion of proximity given there is of geographic location, such as a location defined by a measure such as latitude and longitude. We contend that the definition of location need not fit that traditional mold. Specifically, we may reason about the location of a node by considering the communication connections of each node and its neighbors. We refer to the former notion as geographic-based and the latter as path-based locality.

Our notion of path-based locality is founded the concept of broadcast domain, defined in Section 2. We consider nodes that are in the broadcast domain of another node to be that node's neighbors. Grouping neighbors of neighbors forms neighborhoods. For example,

$$\begin{aligned} &\text{Given:} \quad \mathrm{BD}(x) = \{q, r, s\} \text{ and } \mathrm{BD}(y) = \{r, p\} \\ &\text{Derive} \quad \mathrm{NH}(x, y) = \mathrm{BD}(x) \cup \mathrm{BD}(y) = \{q, r, s, p\}. \end{aligned}$$

We may consider the Most Recent Location (MRL) of Section 2 to have a coarse granularity similar to that of a neighborhood, rather than exact spatial coordinates. Because of this, nodes need not transmit their coordinates via the ad hoc broadcast network. With regard to the subversion problem, an exact location may be less important as we desire to determine only if the node has left the neighborhood.

The difference in these two perspectives (geographic versus connection-oriented location) is seen in the rules used to model them. In the former perspective, the broadcast domain of a node x is defined to be the set of nodes whose location is within the transmission distance of x. In the latter, nodes are defined to be within the transmission distance of x if they are in x's broadcast domain. Fortunately, we easily model both of these locality notions and establish the strongest notion of location as:

$$y \in \mathrm{BD}(x) \iff \big(\mathrm{MRL}(x), \mathrm{MRL}(y)\big) < \mathrm{XR}(x) \ .$$

When considering path-based locality, any node may infer its own location by knowing whom it can communicate with. During routine communications, a node will be able to passively determine which other nodes are in its proximity. A reasonable approximation to the present location can be constructed solely from a node's group peers rather than requiring an enumeration of all nodes within communication range.

Our strong definition of location of a node is illustrated by considering a designated authority to determine if the node is within the neighborhood. In a model-theoretic sense, the designated authority is defined to know the salient relationships between nodes. From a practical perspective, a monitor can detect node engagement in the secure group communications key agreement protocols, and can use that information to construct a set of nodes that that are able to communicate for each group formed.

Given a sufficiently large communication group, the self-location will be very close to the actual location. In our following discussion, we consider the global, model theoretic location to detect a subverted node.

4.2 Detecting Motion

The boundaries of a neighborhood are marked by the set of nodes in a group. There can be overlapping neighborhoods and nodes can simultaneously be members of many groups. The boundary for a neighborhood can be extended as a node moves in a particular direction, but at some point, it will move out of the communication range of the farthest node in the group. The group becomes partitioned, and in the ensuing rekeying process, it is apparent that a node has moved. Over time, the mobile node will leave other groups as well and can be identified and located.

Consider the following example of a neighborhood $\{A, B, C, D, E, F, G\}$ with

$$\begin{aligned}
\mathrm{BD}(A) &= \{A, B, C, D, F\} \\
\mathrm{BD}(B) &= \{A, B, C, D, F\} \\
\mathrm{BD}(C) &= \{A, B, C, D, F\} \\
\mathrm{BD}(D) &= \{A, B, C, D, E, F, G\} \\
\mathrm{BD}(E) &= \{D, E, F, G\} \\
\mathrm{BD}(F) &= \{A, B, C, D, E, F, G\} \\
\mathrm{BD}(G) &= \{D, E, F, G\}.
\end{aligned}$$

Following a motion, the set of nodes that X can communicate with changes. New nodes may come into communication range and distant nodes leave the set.

$$\begin{aligned}
\mathrm{BD}(A) &= \mathrm{BD}(A) - \{B\} \\
\mathrm{BD}(B) &= \mathrm{BD}(B) - \{C, D, F\} \\
\mathrm{BD}(C) &= \mathrm{BD}(C) - \{B\} \\
\mathrm{BD}(D) &= \mathrm{BD}(D) - \{B\}
\end{aligned}$$

$$\mathrm{BD}(E) = \mathrm{BD}(E)$$
$$\mathrm{BD}(F) = \mathrm{BD}(F) - \{B\}$$
$$\mathrm{BD}(G) = \mathrm{BD}(G)$$

Because nodes move during the progression of the battle, this scenario may indicate a subversion only if the tactical battle plan does not call for the asset to relocate. Unexpected motion certainly raises concerns and may lower our confidence in the trustworthiness of the node.

Figure 1 shows our example of two neighborhoods (groups) of nodes. The boundaries of a neighborhood are marked by the set of nodes in a group. Here, nodes A, B, C, E, and F are in the same group and communicate through a common cryptographic key. Likewise, nodes E, D, F, and G are in a group and communicate by using a different key. The shaded circle center at node B represents the effective transmission range of B. In the right diagram, node B is moving away from its group. Note that B moves out of the range of group members C, D, and F thereby reducing its broadcast domain to node A.

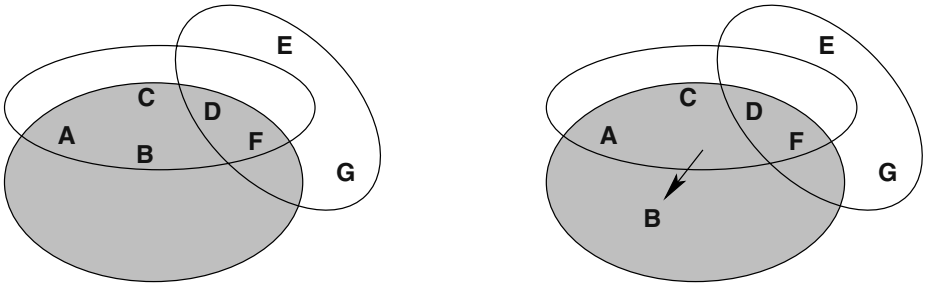


Fig. 1.

Eventually C, D, or F will recognize that B is offline and will initiate a group-rekey operation. At that point, it will become apparent that $\mathrm{BD}(B) = \{A, B\}$ and B is removed from $\mathrm{BD}(C)$, $\mathrm{BD}(D)$, and $\mathrm{BD}(F)$. This change signals that B has moved and if that is unexpected relative to the tactical plan, then B may have been subverted and should no longer be trusted.

5 Conclusion

Ad hoc wireless networks are destined to be an essential element of the battlefield of the future, not to speak of the explosion of their use for personal and industrial applications. In this paper we present a model for reasoning about security characteristics of ad hoc wireless communication protocols. We exercised the model to illustrate a broadcast version of the well-known Diffie-Hellman group key distribution scheme and then developed a new protocol for a special case

situation for battlefield networks employing Silent Partners. Finally, we showed how our model can be used to reason about the mobility of nodes and what this may say about the reliability or trust properties of those nodes.

References

1. G. Ateniese, M. Steiner, and G. Tsudik, "Authenticated Group Key Agreement and Friends", In ACM CCS ACM, November 1998
2. Burrows, M., Abadi, M., and Needham, R. M. "A Practical Study in Belief and Action" In Proc of the 2nd Conf on Theoretical Aspects of Reasoning about Knowledge (Asilomar, Ca., Feb. 1988) M. Vardi, Ed. Morgan Kaufmann, Los Altos, Calif., 1988, pp. 325–342
3. D.R. Cheriton and S.E. Deering. "Host Groups: A Multicast Extension for Datagram Internetworks". In 9th Data Communication Symposium, September 1985
4. R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. "Multicast security: A taxonomy and some efficient constructions", In *Proc. IEEE Infocom,* March 1999
5. W. Diffie and M. Hellman, "New Directions In Cryptography", *IEEE Transactions on Information Theory,* IT-22(6):644–654, November 1976
6. W. Diffie, P. C. van Oorshot, and M. J. Wiener, "Authentication and Authenticated Key Exchanges", *Designs, Codes and Cryptography,* 2(2):107–125, June 1992
7. Michael Fischer and Rebecca N. Wright, "An Efficient Protocol for Unconditionally Secure Secret Key Exchange", Proc of the 4th Symp on Discrete Algorithms (1993), pp. 475–83
8. T. Hardjono, B. Cain, and N. Dorawswamy, "A framework for group key management for multicast security", IETF Internet draft, August 2000. draft-ietf-ipsec-gkmframework-03.txt
9. Lein Harn & Thomas Kiesler, "Authenticated Group Key Distribution Scheme For a Large Distributed Network", IEEE Symposium on Security and Privacy, 1989, pp 300–309
10. Meadows, C., 'A System for the Specification and Analysis of Key Management Protocols'. From 1991 IEEE Computer Society Symp on Research in Security and Privacy, pp. 182–195.
11. C. Meadows and P. Narendran, "A Unification Algorithm for the Group Diffie-Hellman Protocol", WITS (in conjunction with POPL'02), Portland, Oregon, USA, Jan 14–15, 2002
12. M. J. Moyer, J. R. Rao, and P. Rohatgi, "A survey of security issues in multicast communications". *IEEE Network,* pp. 12–23, Nov/Dec 1999
13. Roger M. Needham, Michael D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", CACM, December 1978 vol. 21 #12, pp. 993–999
14. M. Ramalho, "Intra and inter-domain multicast routing protocols: A survey and a taxonomy", *IEEE Communications Surveys & Tutorials,* vol. 3, no. 1, pp. 2–25, First Quarter 2000
15. M. Reiter, "A secure group membership protocol", Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 1994
16. C. Shields and J.J. Garcia-Luna-Aceves, "KHIP—A scalable protocol for secure multicast routing", *ACM SIGCOMM,* 1999

17. M. Steiner, G. Tsudik, M. Waidner, "Diffie-Hellman key distribution extended to group communication", In Proc. 3rd ACM CCS, New Dehli, India, 14–16 May, 1996, pp. 31–7
18. C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs", *IEEE/ACM Transactions on Networking,* vol. 8, no. 1, pp. 16–30, February 2000

Modelling Protocols for Secure Group Communications in Ad Hoc Networks (Transcript of Discussion)

Alec Yasinsac

Computer Science Department, Florida State University.

David Wheeler: Could you clarify one point? Is the group of nodes dynamically updated locally, or is the knowledge shared across the network? In other words, if I sent a message to person X, have I been updated with all the necessary routing information, or is that distributed across the network? It wasn't quite clear which model you were thinking of.

Reply: Thank you very much, and I should have mentioned that. We have intentionally left that undecided. The model, this particular structure that we've provided, doesn't tell you how you're going to fill in the blanks of how this information is accumulated. It may be accumulated, for example, by mobile agents that are traversing your network. It may be accumulated by a floating protocol on some set regular pattern, as someone proposed. There are multiple ways that this information can be accumulated and the model doesn't specify which you have to use, it just allows you to reason, once that information has been accumulated, about what you can do and how you can route.

David Wheeler: Your last sentence has slightly worried me, you can reason about what to do. Now is "you" the group as a whole, or an individual component of it?

Reply: It is "You, the group as a whole." Routing decisions are made locally, so clearly the impact of my last statement of what you, the router, would do, is local. On the other hand, it would also apply to you globally when you are establishing the structure of the network, and the key functionality that you want those nodes to have, and how you want to route. But again, the idea is an abstraction, to allow you a plan based on many different paradigms of accumulating the information from these nodes.

Delegation of Signalling Rights

Pekka Nikander and Jari Arkko

Ericsson Research NomadicLab,
02420 Jorvas, Finland
{Pekka.Nikander,Jari.Arkko}@nomadiclab.com

Abstract. Consider a network of interconnected nodes where each node is identified with a public key. Each node uses the corresponding private key to sign signalling messages. This allows those nodes that know a given node (by its public key) to verify the authenticity of the signalling messages. Under these circumstances, a node may delegate the right to send signalling messages to another node. The delegation may be expressed, for example, in the form of authorization certificates. In this paper we describe how such delegation could be used to optimise signalling paths in mobile and ad hoc network settings. Additionally, we consider the constraints and limitations of the proposed approach.

1 Introduction

Consider a communications network architecture, consisting of a network of nodes, where each node is explicitly identified with a self-generated public key. Furthermore, assume that the topology of the network may change over time, i.e., there may be new connections between arbitrary nodes, existing connections may be broken, new nodes may be introduced, and old nodes may disappear altogether. It may even be possible that the network gets partitioned into two or more partitions over time, and then later on some or all of the partitions join together again.

In such a network, the self-generated public keys act as persistent and secure identifiers of the nodes. They allow, within the limitations of storage capacity, a node to recognize any nodes that is has encountered previously. Furthermore, any node may create signed statements about the other nodes it knows of. These signed statements can be used to express various kinds of information, for example, how the node perceives the other nodes' trustworthiness. However, in this paper we concentrate on the case where a node wants to allow another node to act as a proxy with respect to signalling messages.

For the purposes of this paper, we define the terms signalling and signalling in the context of communication as follows. Consider two (or more) parties that are engaged in communicating with each other. Even though some of the engaged parties may be stateless [1], at least on party must maintain some communication state, and usually all of the participants maintain local state. The purpose of such state is to save bandwidth and enhance resilience by maintaining information about the peers. For example, in the case of the peer(s) of a multi-homed mobile

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 203–214, 2004.

host, the peer(s) keep state about the addresses where the host is currently reachable at. Within this context, signaling is defined as the act of updating the state at the peer(s), and signalling messages carry the necessary information for such updates. For example, such a message may carry information about changes in the set of active addresses.

We focus on how self-generated cryptographic identifiers may be used to optimise signalling traffic resulting from individual nodes moving around, from small subnetworks moving around, and, to a lesser extend, how to apply the method in the case of more generic topology changes. To illuminate its relevance to real world, we first consider separately the cases of individual mobile nodes, small subnetworks having just one connection with the rest of the network, and small subnetworks having more than one connection with the rest of the network. In these three cases, we assume that the structure of the network remains static other than the individual change in focus. Basically, we are able to show that delegation leads to enhanced signalling efficiency without any new security problems. Finally, after these initial exercises, we proceed to generalize the approach and study its limitations.

The rest of this paper is organized as follows. First, in Sect. 2, we discuss most relevant related work. Next, in Sect. 3 we introduce the concept with through a number of basic scenarios, and in Sect. 4 we work towards generalizing the central ideas and discuss some inherent limitations. Sect. 5 contains our conclusions from this research.

2 Related Work

The idea of using self-generated keys to identify functional elements, and to use signatures to delegate rights in the form of authorization certificates is by no means new. Though probably almost as old as public key cryptography itself, the delegation idea was first presented in its modern form in PolicyMaker [2] and at about the same time in the original SPKI/SDSI papers [3,4]. We explored the same ideas in [5], and collected some usage scenarios in [6]. Since then, the ideas have been studied and revised in numerous papers, including [7,8,9].

The idea of representing network nodes by public keys is present in several independent works. One of the earliest works is Radia Perlman's PhD thesis [10], where she describes routing protocols that are robust against byzantine failures. In her work, each node possesses a public key pair and knows the public keys of other trusted nodes. More recently, the Host Layer Protocol / Host Identity Protocol (HIP), currently being discussed at the IETF [11,12,13], uses public keys as primary identifiers. A number of recent proposals for secure ad hoc routing protocols, including Secure Ad hoc On-Demand Distance Vector (SAODV) [14] and Authenticated Routing for Ad hoc Networks [15], are based on the idea of identifying nodes with public keys.

Thus, both the idea of using public keys as primary identifiers for network nodes and the idea of using authorization certificates to delegate rights from a key holder to another are well established. However, to our knowledge the

present work is the first case when someone systematically explores how these methods can be effectively combined to secure signalling messages and what are the limitations of this new approach.

3 Basic Scenarios

In this section we concentrate on a number of topologically simple scenarios that happen to have relatively high practical value, given the current development at the standardization forums. Particularly, we cover the basic cases of individual mobile nodes (Sect. 3.1 and Sect. 3.2), mobile subnetworks (Sect. 3.3), and mixed hierarchical configurations (Sect. 3.4). In the end of the section (Sect. 3.5) we consider how delegations can be combined to achieve even larger performance benefits. Later, in Sect. 4, we consider the more generic case.

3.1 An Individual Mobile Node

The basic need, present in todays mobile phone and mobile IP networks, is to allow a simple end-node to change its point of attachment with a fixed network. Thus, the basic assumptions here are that there is a fixed network, consisting of stationary nodes, and there are one or more mobile nodes that communicate with the fixed nodes. With some additional complications that stem from the possibility of simultaneous movements, it is also possible to extend the model to cover communications between mobile nodes [16].

If each mobile node is identified with a public key, and all states presenting ongoing communications are bound to these public keys, movement itself does not cause any new security problems. The mobile node just informs its corresponding nodes about its new location. The practical implications are discussed in length elsewhere [16], and beyond the scope of this paper.

For the purposes of this paper, the interesting phenomenon is the introduction of the location update messages, needed to update the location state at the corresponding nodes. The moving node sends location update signalling messages to it peers, informing the peers about its new location. These messages allow the peers to learn the new routing address of the moving node, and to redirect their communications appropriately. However, the introduction of these signalling messages opens up new security vulnerabilities. Firstly, if an attacker was able to send false location updates, it could hijack the current communications of a target node. Secondly, a group of a malicious nodes could open up legitimate connections with a number of corresponding nodes, and then – in a concerted act – move the existing packet streams towards a victim node, causing a denial-of-service attack. The victim, which suddenly receives large amounts of unwanted traffic, may choke on that extra traffic [17]. The signalling system can be secured against these vulnerabilities by introducing two countermeasures. Firstly, we require that the mobile node signs the signalling messages so that the correspondent nodes can verify the authenticity of the messages. Since the

ongoing connections are bound to the public key of the mobile node, the corresponding nodes necessarily have the public key in their possession. No external infrastructure is needed. Secondly, we require that before sending any larger amounts of data to the claimed new location, the correspondent nodes must verify that the mobile node is indeed reachable at the said location. This can be achieved by a simple challenge-response protocol. That protects the system against the described distributed denial-of-service attack.

Notice that under this scheme, there is no real difference between mobile nodes that have only one attachment with the fixed network and those that have more than one. If a mobile node has more than one parallel attachments, it can simply enumerate them and inform the corresponding nodes separately about changes. For example, if Alice is currently attached at a location called 10.1.0.1 with her left hand interface, and her right hand interface uses the address 10.2.0.2, she can tell Bob that her left interface is no longer at 10.1.0.1 but at 10.5.0.5.

3.2 The First Optimization

Consider now a case where the link between the mobile node and the fixed network is somehow more expensive than the links within the fixed network. For example, the mobile node may use radio communications while the fixed network is based on wirelines. Furthermore, assume that the mobile node moves within a topologically restricted area. For example, the restricted area may be an access network owned by a telecom operator, as depicted in Fig. 1. In such a case the network may know about the node's movement through some mechanism internal to the access network. Now, since it knows the location of the node, the network would be able to inform the corresponding nodes about the new location of the node. Such a practice would save resources on the high cost link between the network and the mobile node. However, since the location update messages must be signed with the mobile node's private key, the messages sent by the network would not be trusted by the corresponding nodes.

However, if we assume that there is a node in the network that both reliably learns about the new location of the mobile node and that is trusted by the mobile node, the mobile node can delegate the right to send signalling messages to that particular node. For example, if the access network is named by it network prefix, e.g. 10.0.0.0/20, the mobile node can sign Cert. (1).

$$\left\{K^{+}_{MN1}, K^{+}_{MAP}, reg = 10.0.0.0/20\right\}_{K^{-}_{MN1}} \tag{1}$$

Basically, the certificate states that the mobile node, identified by K^{+}_{MN1}, delegates a right to send location update messages within the region 10.0.0.0/20 to a trusted node, the delegate, identified by the public key K^{+}_{MAP}. Given this certificate, the delegate can now send location updates to the corresponding nodes. As long as each location update contains or can refer to this certificate, is correctly signed by the delegate's private key, and the new location is within the given region, the corresponding nodes would accept the update.

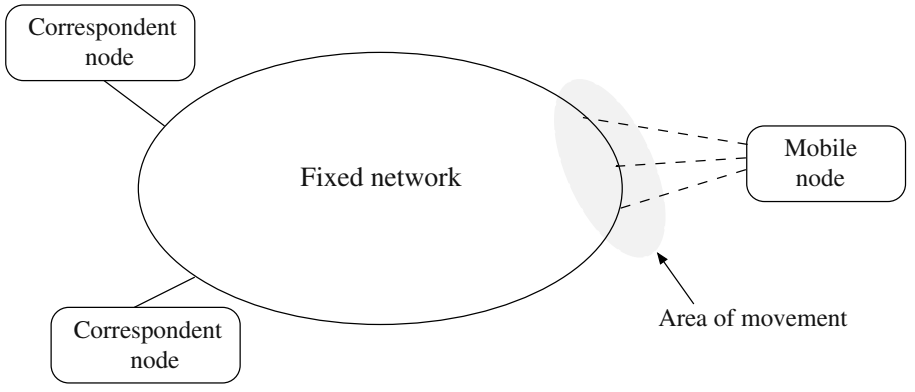


Fig. 1. Restricted movement

It is crucial to understand the reasons why this practice can be considered secure. Firstly, the state at a corresponding node represents information about the mobile node. Consequently, the mobile node can be considered to have a natural right to update this state. Secondly, we assumed that the mobile node trusts the delegate. However, it does not need to trust the delegate in all possible senses, trusting it for correctly sending location updates is enough. Thirdly, the certificate contains the regional limitation (and possibly others), thereby further limiting the risks and the amount and nature of trust needed. And finally, there are no external effects to consider, as we will discuss in Sect. 4.

3.3 A Mobile Network

A mobile network (monet) is a set of mobile nodes that move together. That is, we still assume that there is a fixed network of stationary nodes, and a number of mobile nodes, but now the mobile nodes may cluster together and communicate with the fixed network through a common representative, a mobile router. The scenario is illustrated in Fig. 2. If we do not use delegation, each mobile node must individually inform the corresponding nodes about their movements. On the other hand, the basic delegation scenario is very simple: the mobile nodes just delegate to the mobile router the right to send location updates on their behalf. Since they must trust the mobile router for their connectivity, there is few if any trust problems involved. That is, the mobile nodes are at the mercy of the mobile router, since the mobile router is their only connection to the fixed network. Thus, they have no other choice but to trust the mobile router to convey their messages, and therefore they expose themselves to little new risk if they let the mobile router to act as their signalling proxy.

The reasons for considering this situation secure are the same as in the previous case. The state stored at the corresponding nodes is concerned only about the mobile node(s), creating the natural right of updating it. The certificates can

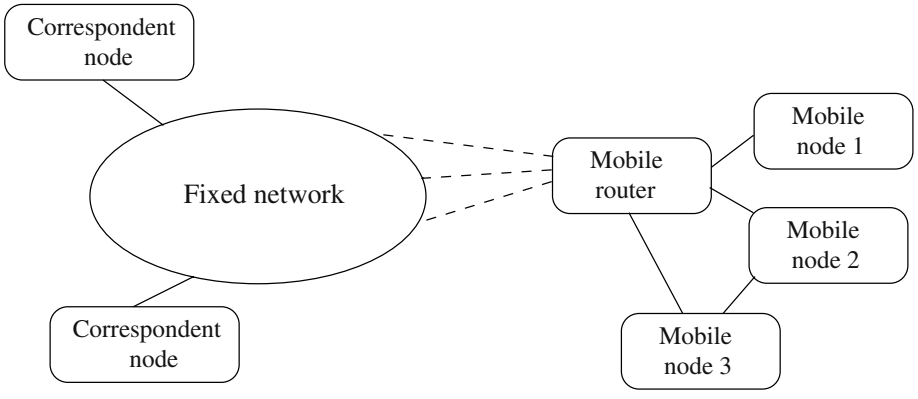


Fig. 2. A mobile network

be designed to limit the delegation to be valid only as long as the mobile nodes stay within the mobile network. Maybe the largest difference is that this time the mobile node needs not to choose to trust the mobile router since it does not have any alternative.

Introducing multi-homing in the sense that the mobile router has more than one attachments with the fixed network causes little problems. However, if the mobile nodes themselves are multi-homed, i.e. simultaneously have connections through the mobile router and through other means, the situation becomes slightly more complex, as discussed next.

3.4 More Complex Hierarchical Scenarios

We now consider a scenario where the mobile nodes form a directed acyclic graph (dag) with respect to their attachment to the fixed network. This is illustrated in Fig. 3. For example, mobile node 2 is able to communicate both directly and through mobile node 1. Similarly, mobile node 5 is able to communicate either through mobile node 2 or through a path created by mobile nodes 3 and 1.

Now it becomes important to make a difference between the different paths. Otherwise an overly eager trusted node between a subject node and the network could claim that the subject node is only reachable through it and not through the other possible path. Consequently, it becomes necessary to name the paths, and to restrict delegations to consider the paths in addition to the nodes and locations.

For example, consider mobile node 2. It has two network interfaces, one forming a communications link with node 1 and another one that is directly connected to the fixed network. The direct connection to the fixed network is named L0, and the link to mobile node 1 is named L1. Thus, the certificate that delegates the right to send location updates concerning L1, given to mobile node 1, would be constructed as as follows.

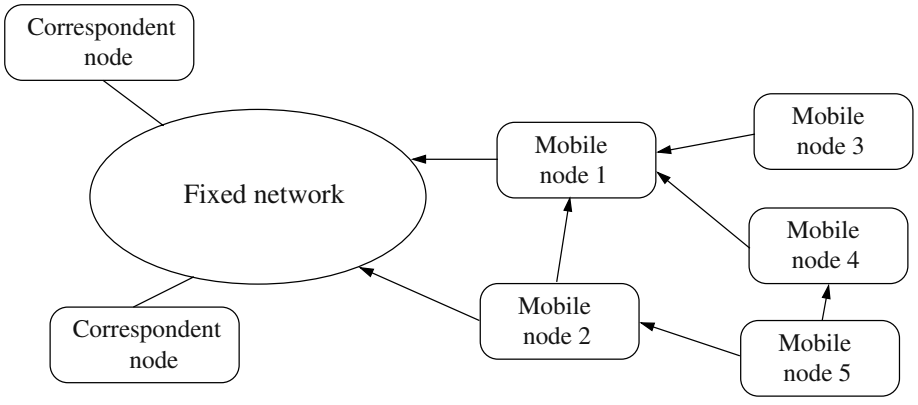


Fig. 3. A directed acyclic graph of mobile nodes

$$\left\{K^{+}_{MN2}, K^{+}_{MN1}, link = L1\right\}_{K^{-}_{MN2}} \tag{2}$$

Given this certificate, MN1 can send location updates to MN2's corresponding nodes, informing them about the new location through which MN2 is reachable, but only with respect to the connection that MN2 has with MN1. Note that the corresponding nodes do not need to have any direct understanding of the nature of the interfaces or links. It is sufficient they can make a distinction between the different names, as delegations concerning a specific interface or link can be changed only by a node that has explicitly been delegated a right to do so. Thus, a named delegation leaves all other rights still intact, and therefore MN2 can independently inform its corresponding nodes about its reachability with respect to its direct connection with the fixed network.

As before, it is useful to analyze the situation in a slightly more detail. The state at the corresponding nodes is still assumed to reflect the connectivity of the mobile node, which is MN2 in this specific case, and therefore the mobile node can be still considered to have a natural right to update that state. However, that state is now split into two distinct parts, one representing L0, the direct connection MN2 has with the network, and the other L1, the connection through MN1. Note that MN1 acts here as a mobile router. Now, since the state has been split, there are two rights that MN2 can delegate, each representing a separate part of the state. In our example, MN2 delegates only the right to update the mobility state associated with L1, but not the part of the state associated with L0. In that way it can remain in full control over L1 while still allowing MN1 to update the location address associated with L1. As discussed above, this limits the risk MN2 is exposed to. That is, MN2 is still able to continue to communicate even if MN1 turns out to be untrustworthy as a router.

3.5 Combined Optimization

The optimisations described in Sect. 3.2 and Sect. 3.4 can now be combined. If MN2 has delegated the right to send location updates concerning L1 to MN1, MN1 may want to further transfer this right to the fixed network side in order to optimise the amount of signalling traffic flowing over the radio link between MN1 and the fixed network. If we allow certificate chaining in the SPKI style, we can combine Cert. (1) and Cert.(2), reducing to

$$\left\{K^{+}_{MN2}, K^{+}_{MAP}, link = L1 \& reg = 10.0.0.0/20\right\}_{K^{-}_{MN2}} \tag{3}$$

Thus, MN2 effectively delegates the right to send location updates concerning its link L1 within the region 10.0.0.0/20 to the trusted delegate at the fixed network. If all nodes behind MN1 do this, all those location updates that otherwise would be sent over the air interface between MN1 and the fixed network can now be generated within the fixed network. In addition to the reduced need for air time, this might also speed up location updates since the access network may be able to send them before the individual nodes would be able to.

The main difference between this and the above scenarios is in decision making. Here MN2 decides to trust MN1, and MN1 independently decides to trust the delegate in its access network. These decisions can be securely chained since MN2's decision delegates all power to update state representing L1 to MN1. Thus, MN1s decision of further delegating the right to the delegate is within the power it received from MN2.

4 Generalization and Limitations

In all the previous scenarios, a node delegated the right to send signalling messages to another node, a delegate node. The main purpose of this practice is to optimize, i.e., to gain advantages in terms of time, energy, etc. Consequently, the delegate must be topologically located between the delegator and the correspondent node, or otherwise there would be no or little advantaged to gain. Additionally, the delegate must know the changes in the delegator's state so that it can send signalling messages in the first place. For the scheme to be secure, the delegator must be able to determine the scope within which it can trust (willingness to trust) and must trust (necessary trust) the delegate. For example, in the second scenario, the mobile node has no other choice but to trust the mobile router; it cannot communicate otherwise, while in the third scenario it was able to mitigate it risk level by introducing separate pieces of remote state and separate rights for managing each of those pieces. As we consider the method's applicability to more generic situations, we have to keep these principles in mind.

In the rest of this section, we consider some of the practical consequences.

4.1 Byzantine Routing

In the case of Byzantine routing [10], the nodes need to rely on each other for carrying traffic. Consequently, the nodes must collectively maintain an idea of

who are trustworthy and who are not so trustworthy. Even though this does not require delegation to work, the situation can be considered as an generalization of the scenarios presented above. Thus, it looks reasonable to assume that delegation might bring efficiency gains. The main problem seems to lie in deciding whom to trust enough. Another problem corners around the implicit liability that the nodes have towards all those nodes whose traffic they pass. That is, a node acting as a router may be considered to have taken the burden of spending an amount of resources to pass the packets towards the destination. A limitation must be place on the resource consumption in a reasonable way in order to remain trustworthy but not to exhaust oneself. However, the detailed study of that is beyond the scope of this paper.

4.2 External Effects

The main body of this paper has considered signalling in a context where the signalling messages are used to maintain internal, externally invisible communications state. However, there are other types of signalling protocols where the signalling messages may cause externally noticeable effects. For example, in IP based telephony the Session Initiation Protocol (SIP) is used to open new audio or multimedia connections.

When external effects are possible, care must be taken to ensure that the delegated rights are authorized in the first place. That is, the rights may not naturally fall upon the parties, as in the case of mobility and multi-homing signalling, but the rights must be explicitly assigned to a party before they can be delegated. On possible way to achieve that is to apply an explicit security infrastructure on the top of the identifier keys, and use manually created authorization certificates to assign the initial rights.

5 Conclusions

In this paper we have discussed how delegation, based on the idea of authorization certificates, can be used to securely optimise signalling paths in a network where the network nodes are identified with public keys. The idea seems to be applicable to a wide variety of practical problems, including mobile subnetworks and ad hoc networks. However, once external effects or additional nodes enter the picture, a number of limitations must be considered. However, we surmise that the presented approach is applicable also to other applications, as long as the limitations are concerned.

Acknowledgments. We want to express our gratitude to our colleague Jukka Ylitalo, who innocently throw up the original idea, which we then deepened and extended. We are also grateful for Ilkka Uusitalo for his comments on early versions of this paper.

References

1. Tuomas Aura and Pekka Nikander, "Stateless connections", in proceedings of International Conference on Information and Communications Security ICICS'97, Beijing, November 1997, pp. 87–97, Lecture Notes in Computer Science 1334, Springer Verlag 1997.
2. M. Blaze, J. Feigmenbaum and J. Lacy, "Decentralized Trust Management". IEEE Conference on Security and Privacy, Oakland, CA. May 1996.
3. Carl Ellison et. al., "SPKI Certificate Theory", RFC 2693, IETF, September 1999.
4. Ronald L. Rivest and Butler Lampson, "SDSI – A Simple Distributed Security Infrastructure", published several times, 1996, http://theory.lcs.mit.edu/~ rivest/sdsi10.html
5. Ilari Lehti and Pekka Nikander, "Certifying trust, in Proceedings of the Practice and Theory in Public Key Cryptography", (PKC)'98, Yokohama, Japan, Springer-Verlag, February 1998.
6. Pekka Nikander, "An Architecture for Authorization and Delegation in Distributed Object-Oriented Agent Systems", Ph.D. Dissertation, Helsinki University of Technology, March 1999.
7. Martín Abadi, "On SDSI's linked local name spaces", in Proc. 10th IEEE Computer Security Foundations Workshop, pages 98–108, Rockport, MA, June 1997. IEEE Computer Society Press. http://citeseer.nj.nec.com/abadi98sdsis.html
8. Jon Howell, David Kotz, "A Formal Semantics for SPKI" , in the Proceedings of 6th European Symposium on Research in Computer Security (ESORICS 2000), October 4–6, 2000, Toulouse, France, LNCS 1895, pp. 140–158, Springer, October 2000.
9. Yki Kortesniemi, Tero Hasu, and Jonna Särs, "A revocation, validation and authentication protocol for SPKI based delegation systems", in Proceedings of the 2000 Network and Distributed System Security Symposium (NDSS 2000), pp. 85–101, San Diego, California, February 2000.
10. Radia J. Perlman, "Network layer protocols with Byzantine robustness", Ph.D. Thesis, Massachusetts Institute of Technology, 1988.
11. Robert Moskowitz, "Host Identity Payload Architecture", work in progress, Internet Draft (expired), February 2001, http://klovia.htt-consult.com/draft-moskowitz-hip-arch-02.txt
12. Robert Moskowitz, "Host Identity Payload and Protocol", work in progress, Internet Draft draft-moskowitz-hip-05.txt, November 2001, http://klovia.htt-consult.com/draft-moskowitz-hip-05.txt
13. Robert Moskowitz, "Host Identity Protocol Implementation", work in progress, Internet Draft (expired) draft-moskowitz-hip-impl-01.txt, Feb 2001, http://klovia.htt-consult.com/draft-moskowitz-hip-impl-01.txt
14. Manel Guerrero Zapata, "Secure Ad hoc On-Demand Distance Vector (SAODV) Routing", unpublished manuscript, sent to the manet mailing list, 08 Oct. 2001, ftp://manet.itd.nrl.navy.mil/pub/manet/2001-10.mail
15. Bridge Dahill, Brian Neil Levine, Elizabeth Royer, and Clay Schields, "A Secure Routing Protocol for Ad Hoc Networks", Technical Report UM-CS-2001-037, University of Michigan, August 2001.
16. Pekka Nikander, "A Case for the Host Identity Payload: An Architecture for Multi-Homed Mobile Hosts", unpublished manuscript submitted for consideration to be published at Mobicom 2002, Ericsson Research, March 2002.
17. T. Aura and J. Arkko, "MIPv6 BU Attacks and Defenses", work in progress, Internet Draft draft-aura-mipv6-bu-attacks-01.txt, February 2002.

Delegation of Signalling Rights (Transcript of Discussion)

Pekka Nikander

Ericsson NomadicLab

Laurent Eschenauer: I like this a lot, because you can secure routing with these kinds of ideas, especially source-routing. For example, DSR can be secured if you have a secret association between the two endpoints, if you know the public key of the destination. This can give you the public key because the public key is the address you are going to reach, so you can make the link between two.

The other thing I wanted to say is that usually I have the feeling that we have a sort of loop between signalling or routing and security. It is as if security needs the routing and the signalling, and the routing and signalling need the security. This approach could really break up that deadlock by having a sort of weaker security; just a weak notion of identification at a low level, upon which you build the routing and the signalling, and over which you can build the real secure infrastructure. Personally, I would say that you are possibly going too far talking about trust, and management of trust, at that level. I would really keep it low level and try to support the routing and the signalling, and then build a real security infrastructure above that.

Reply: Well it is possible to build some kind of infrastructure upon this, so you can always have certificates, or whatever, that say more about these public keys. There is no problem with that. I'm more interested in what you can do without having that, or maybe having just fragments of that kind of public key infrastructure. For example, you could have one lot of public key infrastructure here and another lot of public key infrastructure over there, but no real official connection between them.

Virgil Gligor: So essentially this is the base layer for bootstrapping trust. You need it in a lot of environments, for example in networks of sensors, you need to bootstrap trust in some sense, and find neighbours, and start sharing your keys. But as for the external effects on your network, it seems that we inherit a lot of trouble when we bring the notion of trust from the real world into the network. We don't have the same difficulties when we bring names into play because we have the binding which identifies the addresses. Bringing the external environment into the network is the challenge.

Ernie Cohen: As long as you're just concerned with signalling and things like that, the signatures would be OK. If you just wanted signatures, it could also be done without a public key at all, just using hashes, which might be somewhat more efficient.

Reply: Can you do delegation without public keys?

Ernie Cohen: If you sign it, it's just a matter of being careful about how efficient the checking is.

Reply: The real issue here is that I'm assuming you don't know the peers and you don't necessarily have a physical conduit for sharing a private key.

Ernie Cohen: But you're assuming that you see their public keys, so it's the same problem.

Bruce Christianson: When you sign something without using a public key, how do you verify that without being able to forge their signature?[1]

Ernie Cohen: Well typically you have to follow the hash chain. The main limitation on your ability to do this is that you have to be careful to make the chain to be verified pretty short. This only works as long as you're talking about something where people aren't going to be putting in tons and tons of effort to try to break your signal.

George Danezis: Actually we saw an application of the idea of signing the packets using a hash chain, and you just need the other nodes to know the first hash in the chain.

Richard Clayton: What you described is basically OK as far as it goes, but it is the next bit that is difficult because if you're saying you recognise people by the public key, then you have to accept that malicious people will give their public key to millions of other people. If you're using that, in any sense, to allow access to a resource then that is a problem.

The other thing is that you don't know for sure that everybody else in the network isn't the same person. Maybe there's only two of you there. From your point of view there are a million other people, they all have different numbers, they're all signing each others keys and saying that they're all terribly trustworthy yet, in reality, they are all the same person.

Reply: Right, of course. That was the reason why I said before, that maybe you always want to ensure that there is some part of trust based on personal, or maybe delegated, trust from yourself to such a cloud. I'm suggesting that maybe you could have some kind of heuristic to create that kind of trust. If a single person creates a cloud of a billion keys, cross certifying each other, maybe we could have some heuristic that says that that's probably a hoax: I'm not going to trust that cloud because the connectivity between that cloud and the rest of the world is so sparse, or something like that.

Richard Clayton: But once the cloud becomes big enough then you can almost certainly fool some innocent people into joining it because they are not as cautious as you are, and immediately you have a problem. It is like the eBay idea, where your trust is based on your past experience, and we've seen a number of times this doesn't really work very well at all in the long term.

Reply: I agree there are limitations here. That's why we have these workshops. Maybe we could quantify those limitations and somehow work out what models would be needed for that kind of quantification. But this approach is what I'm interested in.

[1] See for example LNCS 2133, pp 182–193.

Mobile IPv6 Security

Tuomas Aura

Microsoft Research Ltd.
Roger Needham Building, 7 JJ Thomson Avenue, Cambridge, CB3 0FB, UK
tuomaura@microsoft.com

Abstract. This paper presents a case study of security protocol design: authentication of binding updates in Mobile IPv6. We go step by step through the threat analysis and show how each threat is addressed in the protocol design. The goal is to solve any new security issues caused by the introduction of mobility without requiring any new security infrastructure.

1 Introduction

This paper describes the Mobile IPv6 security protocol. The focus is on the authentication of binding updates, i.e. location information sent by the mobile to its correspondents.

We explain the security threats created by the introduction of mobility and the mechanisms that have been used to prevent the attacks. The protocol design is unusual and it would not be considered secure by the measures of traditional security protocol analysis. The security of the protocol depends on the partial reliability of the Internet routing infrastructure. The reason is that the protocol must work between any mobile node and any other Internet node that have no previous relationship, and we cannot assume the existence of a global PKI or other global security infrastructure. On the other hand, the only security requirement was to counter the new threats created by mobility. The protocol does exactly that. This is a pragmatic way of thinking when introducing new technology such as mobility.

The return-routability protocol described in this paper was originally a part of a protocol family designed by Michael Roe, the current author, Greg O'Shea and Jari Arkko [10]. Many of the protocol details have since been refined by the IETF Mobile IP working group [4]. The solution enabled the Mobile IPv6 standardization process to continue after it had halted because of security concerns. When designing the protocol, we discovered a new class of attacks and introduced new defense mechanisms that have since been copied to other mobility protocols. The protocol described in this paper is a slightly simplified version

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 215–234, 2004.

of the actual Mobile IP6 protocol. We concentrate on the binding-update messages sent by the mobile to its correspondents and ignore some details that are intended to protect messages sent in the other direction.

The paper is organized roughly to follow the design process. We first introduce the Mobile IPv6 protocol and route optimization in Section 2. Section 3 describes the basic binding-update authentication protocol. Section 4 explains how even authenticated binding updates can be used for denial-of-service attacks and how these attacks are prevented. Section 5 explains some less serious threats and how the protocol was enhanced to mitigate them. Section 6 concludes the paper.

2 Mobile IPv6

Mobile IPv6 is an IP-layer mobility protocol for the IPv6 Internet. It is being standardized by the IETF. The idea is that when mobility, like any other functionality, is implemented in the network layer, it needs to be implemented only once and will then be transparently available for all higher-layer protocols. It remains to be seen how well this promise is fulfilled in practice. There are, however, some applications like mobile VPN access, for which Mobile IP is clearly a good solution. This section describes the Mobile IPv6 architecture and protocol.

2.1 The Mobile Network Architecture

The first half of an IPv6 address indicates the subnet to which the address belongs and it is used for routing IP packets across the Internet. Thus, when a mobile Internet host (called *mobile node (MN)* in the Mobile IPv6 terminology) moves to a different place in the network topology, its subnet and, thus, IP address necessarily change. This creates two kinds of problems: existing connections (e.g. TCP connections and IPSec security associations) between the mobile and other hosts (called *correspondent nodes (CN)*) become invalid, and the mobile is no more reachable in its old address for new connections. The former problem is important in stateful protocols and has little effect of stateless protocols such as HTTP. The latter problem typically concerns servers and not client computers.

Mobile IPv6 has two basic goals: all transport-layer and higher-layer connections and security associations between the mobile and its correspondents should survive the address change, and the mobile host should be reachable as long as it is connected to the Internet somewhere in the world.

Mobile IP makes some quite strong assumptions about the environment in which it is used. First, all mobile hosts have a home network and a *home address (HoA)* on that network. This is a reflection from a time when mobility was an

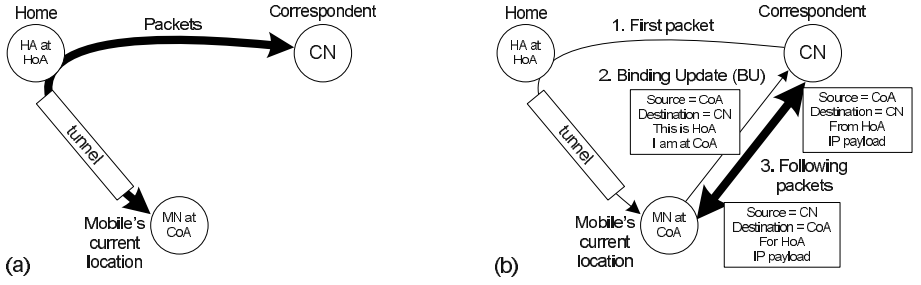


Fig. 1. Mobile IPv6 tunneling (a) and route optimization (b)

exception: few Internet nodes would be mobile and even they would for most of the time remain stationary at home. In any case, Mobile IP solves the reachability problem by ensuring that the mobile is always able to receive packets sent to its home address.

The IP address of a stationary IP node normally serves two purposes: it is both an identifier for the node and an address that is used for routing messages to the node. Mobile IP preserves this dual use of home addresses. The home address is an identifier for the mobile, as well as an address to which correspondents can send packets. The mobile's current location, called *care-of address (CoA)*, on the other hand, is a pure address and serves no identification purposes.

Any IPv6 address can be or become mobile and there is no way of distinguishing a mobile and stationary host by just looking at its address. This is because the Mobile IP protocol was originally designed to be transparent to the mobile's correspondents and the correspondent did not need to know that the mobile, in fact, was a mobile.

2.2 The Mobility Protocol

The transparent mode of operation is shown in Figure 1(a). At its home network, the mobile has an agent, called *home agent (HA)*. The home agent is a router that tunnels packets to and from the mobile. It intercepts packets sent by correspondents to the mobile's home address and forwards them to the mobile at its current location over an IPIP or IPSec tunnel. When the mobile wants to send packets to a correspondent, it sends them to the home agent over the reverse tunnel. The home agent un-encapsulates the packets and forwards them to the correspondent. When the mobile moves to a new location, it tells the home agent its new care-of address.

The tunneling protocol is sufficient to enable mobility but it results in suboptimal routing. Packets between the mobile and its correspondents have to travel

via the home network, which may be far away. To rectify this problem, Mobile IPv6 defines a mechanism called route optimization. The optimization requires changes to the correspondent but it is seen as so important that every IPv6 host has to support the protocol.

Route optimization typically works as shown in Figure 1(b). When the mobile receives a tunneled packet, it initiates the route optimization protocol. The mobile sends to the correspondent a message called *binding update (BU)*. The binding update contains the mobile's home address and current care-of address. The correspondent stores this information in its binding cache, which is effectively a routing table: it tells that packets destined to HoA should instead be sent to CoA. Finally, the correspondent acknowledges the binding update. (For simplicity, the acknowledgements are not shown in the figures and we ignore the authentication of messages from the correspondent to the mobile.)

The route optimization is a voluntary in the sense that either the mobile or the correspondent can refuse to do it and they can continue communicating via the home agent. It is, however, and important optimization because sending packets via the home agent can be very inefficient.

After the binding update, the following packets between the mobile and the correspondent are sent directly. The packets from the mobile to the correspondent contain a header field called *home address option (HAO)*, which tells the correspondent that although the source address is CoA, the packet is actually from the node whose address is HoA. The packets from the correspondent to the mobile contain a *routing header*, which tells the mobile that although the packet is destined to CoA, it is really intended to HoA. The headers are, in effect, a kind of tunnel between the mobile and the correspondent. Every few minutes, the mobile needs to send another binding update to refresh the binding cache entry even if the care-of address has not changed.

The assumptions and design choices made in the Mobile IP protocol are not necessarily the same that would be made if the Internet mobility protocol were designed again from scratch. Nevertheless, these are the protocol and assumptions that we had to live with when considering Mobile IPv6 security.

3 BU Authentication

It is quite obvious that the binding update protocol, if implemented as described above, would create serious new security vulnerabilities. The first thing that one notices is that the binding updates are not authenticated. This section describes the basic attacks using unauthentic BUs and a BU authentication mechanism.

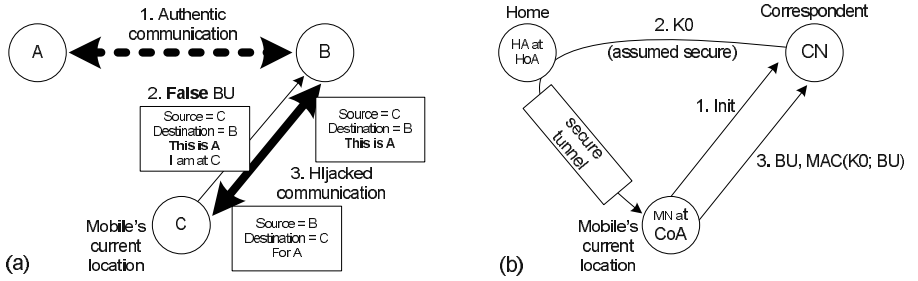


Fig. 2. False BU (a) and routing-based authentication protocol v.1 (b)

3.1 The Need for Infrastructureless Authentication

A possible attack is shown in Figure 2(a). An attacker at address C sends a false binding update to an Internet node B, claiming to be a mobile with home address A. If B, acting in the role of a correspondent, believes this binding update, it will redirect to C all packets that are intended for A. Thus, the attacker can intercept packets sent by B to A. The attacker can also spoof data packets from A by inserting a false home-address option in them. This enables the attacker to hijack existing connections between A and B, and to open new ones pretending to be A. The attacker can also redirect the packets to somewhere else than C, which prevents A and B from communicating with each other. End-to-end data protection, e.g. IPSec or SSL, prevents most of the attacks but not denial of service (DoS).

These attacks are serious because A, B and C can be any IPv6 addresses anywhere on the Internet. All the attacker needs to know is the IPv6 addresses of A and B. Since there is no visible difference between a mobile home address and a stationary IPv6 address, the attacks work as well against stationary Internet nodes as against mobile ones. The possibility of these attacks caused IETF to halt the Mobile IPv6 standardization process until a solution for authenticating binding updates was found. It is believed that deployment of the protocol without security could result in a break-down of the entire Internet.

Obviously, the solution is to authenticate the binding updates. A typical authentication mechanism would involve a trusted online server or a public-key infrastructure (PKI). The problem is that the authentication needs to work between any mobile Internet node and any correspondent. There does not currently exist any authentication infrastructure that could be used for such global authentication between any two IPv6 nodes. Neither is it realistic to suggest creating such infrastructure for the needs of Mobile IPv6. Hence, using the conventional authentication mechanism would confine route optimization to intra-organizational use where the required security services are in place.

For the above reason, we were forced to consider unconventional authentication methods. The advantage we had on our side is that the security requirements for BU authentication are unusually weak. The stated goal in the IETF working group was that the Mobile IPv6 protocol should be at least as secure as the current non-mobile IPv4 Internet. This means that we were not confined to designing a traditional strong security protocol. Our ambition was limited to making sure that Mobile IPv6 does not introduce any new major vulnerability to the Internet. The goal was not to create a strong general-purpose authentication protocol.

The IP layer provides two kinds of infrastructure. First, the addressing architecture [2] provides Internet nodes unique, globally routable IPv6 addresses. Second, the routing infrastructure [3] delivers packets across the Internet to their destination address. It turns out that both the addressing and the routing can be used to bootstrap some form of authentication, not necessarily as strong as a PKI would enable in closed networks but nevertheless better than no authentication. Since these techniques do not require any special security infrastructure, they are, somewhat misleadingly, called *infrastructureless authentication*.

The address-based technique was first proposed in the CAM protocol for binding update authentication by O'Shea and Roe [8]. The basic idea was to create the second half of the home address as a one-way hash of the mobile node's public key. Such addresses are called cryptographically generated addresses (CGA). The mobile signs the binding update and attaches its public key to the message. The correspondent can verify without any additional infrastructure that the binding update was signed by the owner of the home address. While the authentication of the sender's IPv6 address would be of little value in most applications, it is exactly what is needed to authorize the binding update. There were several further proposal for using this technique for protecting mobility protocols [6,5].

The routing-based authentication will be covered in detail in the rest of the paper. Our original protocol design allowed both types of infrastructureless authentication but the IETF working group chose to standardize only the simpler routing-based technique.

3.2 Routing-Based Authentication

The second infrastructureless authentication method is based on the fact that routing in the Internet is semi-reliable. It is difficult for a remote attacker to change the route of packets that do not travel via the attacker's network. Thus, in order to sniff or intercept a packet, the attacker needs to be on its route.

The first version of the BU authentication protocol is shown in Figure 2(b). The idea is that after the mobile initiates the BU protocol (message 1), the cor-

respondent sends a secret key as plaintext to the mobile's home address (message 2). The home agent intercepts the message and forwards it to the mobile via a secure tunnel. The mobile then uses the key to compute a message authentication code for the binding update (message 3). This mechanism is called *return-routability test for the home address* because the mobile must return to the correspondent (a function of) a value sent by the correspondent to the home address. In effect, the correspondent verifies that the mobile is able to receive messages at the home address.

In order to break the protocol, i.e. to spoof a binding update, the attacker needs to be on the route between the correspondent and the mobile. Thus, the protocol is not secure against the standard attacker model where the attacker can sniff and intercept all messages on the network. It is natural that most readers previously unfamiliar with the protocol will at this point object to the idea of sending a key in plaintext. There are, nevertheless, strong arguments in favor of the design.

First, the number of potential attackers and targets is dramatically reduced. Without authentication, any Internet node C could spoof binding updates from any Internet node A to any Internet node B. In our protocol, the attacker C must be on the route from B to A, which means that there are typically only tens or hundreds of nodes that can execute that attack, i.e., the routers between A and B and the hosts on the local networks of A and B. On the other hand, a malicious node is able to target only the connections that pass though it its local network. For a typical attack, such as a compromised host, the number of such connections is small. This reduction in the scale of the potential damage alone means that deployment of the Mobile IPv6 would no longer be a danger to the Internet's stability.

Second, the protocol fulfills the explicit design goal of being as secure as the current Internet without mobility. Assume that the mobile node never leaves its home network and always communicates directly from the home address. In that case, an attacker on the route between A and B can spoof, intercept and sniff packets between them, and it can execute all the same attacks that were possible by exploiting the weaknesses of our BU authentication protocol. Therefore, we argue that the simple protocol of Figure 2(b) is sufficient for authenticating the sender of a binding update.

The way the mobile and the home agent establish the secure tunnel between themselves is beyond the scope of this paper and currently depends on the implementation. We assume that the mobile and the correspondent are in a close alliance and, thus, have a pre-shared key or another method for authenticating each other. Similarly, the mobile can send binding updates to the home agent via a secure tunnel, so that the home agent knows where to forward the packets. (This is theoretically straightforward but relatively cumbersome in practice because some authentication protocols use the source address of packets as the endpoint identifier.) On the other hand, if the mobile and the home agent do not

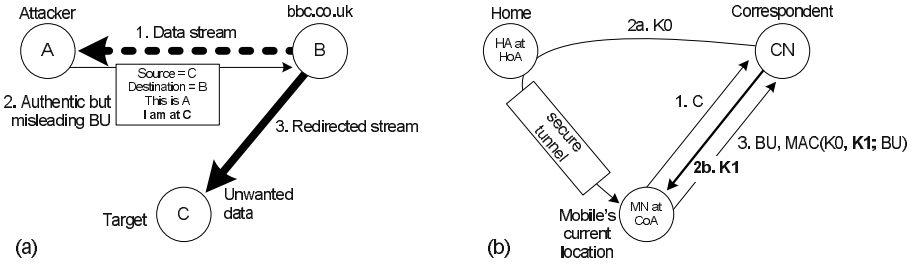


Fig. 3. Bombing attack (a) and verifying the care-of address (b)

have a secure mechanism for authenticating each other or if they do not trust each other, then the assumptions of our protocol do not apply and some other kind of protocol is needed.

4 Verifying the CoA

The protocol described above is sufficient to authenticate the sender of the binding update and, thus, solves the problem that we started with. In the process of designing the protocol, we had to develop an in-depth understanding of the threats against Mobile IPv6. As a result, we discovered another type of treat that is at least as serious as spoofed binding updates. This section explains how even authenticated binding updates can be used to amplify a packet flooding attack, and how the protocol was modified to prevent the attack.

4.1 Bombing Attacks

The key observation is that a binding update contains two pieces of information, HoA and CoA, and the protocol described above only verifies the correctness of the HoA. Even if the binding update is authentic in the sense that it was sent by a mobile node whose home address is the one stated on the packet, the mobile might provide a false value for the care-of address. In other words, the mobile may be lying about its own location.

Once we made the above observation, it is easy to come up with an attack. Figure 3(a) shows a scenario where the attacker A tricks a public web site B into sending a flood of unwanted packets to a third party C. The attacker A first starts to download a stream of data, such as a long TCP stream, from a public server B. It then sends an authenticated binding update to the server claiming to be at the care-of address C. The server accepts the binding update because A used an authentic home address. (A does not need to be mobile. It can use its

own stationary address A as the home address and act both as the home agent and as the mobile node in the binding update protocol.) As a result, the server redirects the data stream to the false care-of address C.

Readers familiar with communications protocols will have noticed by now that B will soon stop transmitting the data stream because it does not receive acknowledgments from C. Unfortunately, this does not help much because the attacker can spoof the acknowledgments. Since the attacker received the first packets of the data stream, it knows the initial TCP sequence numbers and can spoof TCP acknowledgments. Moreover, the attacker only needs to send one acknowledgment per TCP window, which means that by spoofing only a few packets it can get B to send a large data stream to C.

Alert readers might also note that the recipient of unwanted TCP packets usually sends a TCP Reset signal to the source of the packets, which puts in immediate stop to the data stream. Thus, one might assume (as we did for quite a while) that the target node C send a TCP Reset signal to B. Unfortunately, this does not quite work in our case. The packets sent by B to C have a routing header that says the packets are intended for A. When the IP layer in C's stack processes the routing header, it encounters a strange address A and drops the packet without ever processing the following TCP header. Thus, no TCP Reset will ever be sent.

The attack is serious because it can be used to bomb any Internet node with data and the target node cannot do anything to protect itself. If used in combination with distributed denial-of-service (DDoS) attacks, the bombing attack can cause serious problems to the stability of the Internet.

4.2 A Bombing-Resistant Protocol

Out first reaction to the discovery of the bombing attack was that, before sending data to the new care-of address, the correspondent should somehow verify that the mobile is in that address. That is, the correctness of both the HoA and the CoA in the binding update needs to be checked. But secure verification of the physical location of a mobile node turns out to be extremely difficult, especially when the only thing the correspondent knows about the mobile is the home address, and when the access network does not participate in the protocol at all. The solution is to relax the requirements a bit, as we'll explain below.

Figure 3(b) shows how we the improved protocol. The correspondent sends a second secret key to the mobile. This key is sent directly to the mobile's care-of address. The correspondent uses both keys to compute the MAC on the binding update. This proves to the correspondent that the mobile is able to receive messages sent to the new care-of address. This mechanism is called *return-routability (RR) test for the care-of address.*

The above protocol does not strictly prove that the mobile is located at the new care-of address. The attacker could capture the second key at the CoA or on the route from the correspondent to the CoA. But if that is the case, the stream of unwanted packets will flow to or through the attacker's network. Thus, the attacker will suffer from the bombing as much as the target node, and the attacker could just as easily send the flood of packets itself, rather than using the correspondent as the sender.

Effectively, the correspondent in the improved protocol verifies that someone on the new route wants to receive the data. This is sufficient to make packet-flooding attacks unattractive. (The reader may have noted that the return-routability test for the CoA is similar to transport-layer acknowledgment and the same effect could have been achieved by enhancing the transport-layer acknowledgement mechanisms.)

Although we have used the terms authentication and verification, both of the return-routability tests can be seem as forms of authorization. The RR test for the HoA authorizes the sender of the binding update to change the binding for the home address. The RR test for the CoA authorizes the sender to redirect data to the care-of-address. These are quite different security goals and we have achieved them using curiously symmetric mechanisms. This is perhaps best explained by viewing the two tests as a way to verify that the sender is authorized to control the use of the two addresses.

5 Other Attacks

Verification of the home and care-of addresses is sufficient to prevent most attacks that exploit weaknesses of the Mobile IPv6 route optimization. The return-routability protocol does this and, thus, protects the Internet from the new vulnerabilities introduced by the mobility mechanism.

But like in all security protocols, there are a number of potential attacks against the security protocol itself that need to be considered. In this section, we make small changes to the protocol of Figure 3(b) to prevent state-storage exhaustion and reflection attacks. We also describe a class of attack that cannot be prevented by any security protocol.

5.1 Stateless Correspondent

It is well-known that stateful protocols expose the protocol participants to denial of service attacks. In particular, if a host stores a state in a protocol run that someone else has initiated and before authenticating the initiator, an attacker can initiate the protocol many times and cause the host to store a large number of unnecessary protocol states.

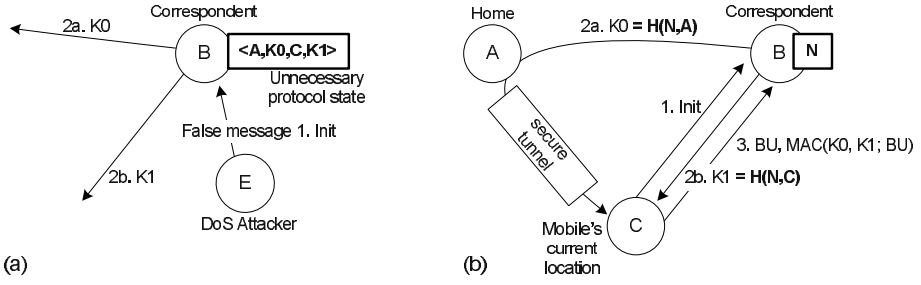


Fig. 4. State-storage exhaustion attack (a) and stateless correspondent (b)

Figure 4(a) shows how this attack works against our protocol. The attacker sends a spoofed initial message with a false home address A and false care-of address C. The correspondent responds with two randomly generated secret keys, which it has to remember until it receives the authenticated BU. If the attacker repeats this many times, the correspondent will not be able to store all the state data and may drop some initial messages. This may prevent legitimate mobiles from using route optimization with the correspondent.

While this attack is not nearly as serious as the one described earlier and it could be prevented by adding memory and managing the state storage carefully, it is much easier to design the protocol to be stateless. (For stateless design techniques, see e.g. [1]). In Figure 4(b), the correspondent does not store a separate state for each mobile. Instead, it stores a single periodically-changing randomly-generated master secret N and computes the two secret keys K0 and K1 with a one-way hash function from the master secret and from the addresses (HoA and CoA). The correspondent does not remember the keys but instead recomputes them when it receives the authenticated binding update. This means that the correspondent can remain stateless until it has authenticated the mobile.

The attack is similar to the SYN flooding attack against the TCP protocol [11]. The two messages sent by the correspondent to the mobile's home address and care-of address are similar to the SYN cookies that have been used to prevent the SYN flooding attack.

Careful readers might suggest two further improvements: binding the two secret keys (K0 and K1) together so that keys from different protocol runs cannot be mixed and matched, and making the two key formats different so that K0 cannot be replayed in the role of K1 and visa versa. While both improvements are unnecessary, they might increase the robustness of the protocol if the assumptions or goals change in the future. The idea of binding the keys together

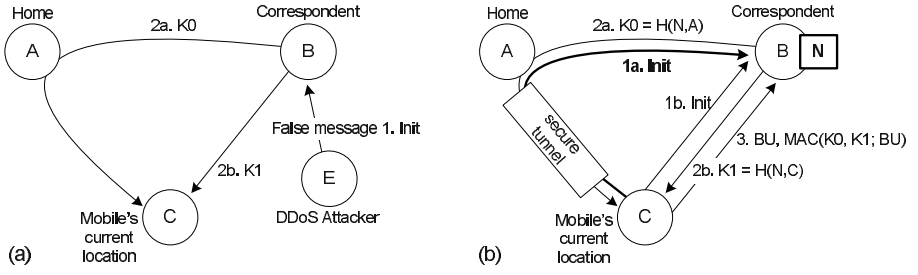


Fig. 5. Reflection attack (a) and balanced messages (b)

was rejected because it is useful to be able to reuse K0 within a short time even if the care-of address changes. The idea of making the key formats slightly different, on the other hand, was adopted to the standard protocol. This is done by inserting octets 0 and 1 into the respective hash inputs.

6 Preventing Reflection and Amplification

Another attack that takes advantage of the security protocol is shown in Figure 5(a). The attacker E spoofs the initial message, which induces the correspondent to send two messages to the mobile. This causes two problems. First, the attacker sends only one packet but two arrive at the mobile. Thus, the attacker can use the binding-update authentication protocol to amplify a packet flooding attack against a mobile node by a factor of two. Second, the two messages arriving at the target of the flooding attacks have the correspondent's address as their source address. Any efficient mechanism for tracing the source of the attacks probably won't be able to trace the attack back to its real origin. (For a detailed discussion of the problems caused by reflection, see [9].)

While these attacks may not seem very serious, it is hard to justify a security protocol that creates new vulnerabilities. The problem was solved by duplicating the initial message. In the protocol of Figure 5(b), the mobile sends one initial message via its home agent and another one directly to the correspondent. The correspondent responds to both initial messages independently by sending a secret key to the same address that the initial message came from. The mobile needs to send both initial messages to receive both keys. The result is that the correspondent sends only as many messages as it receives, thus eliminating the amplification problem, and the correspondent always responds to the same address from which the initial message appears to come, which may make it easier to trace the origin attack using standard methods.

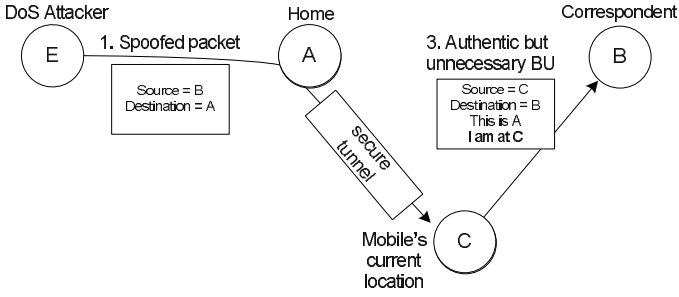


Fig. 6. Inducing unnecessary BUs

7 Avoiding Unnecessary Authentication

There is one last attack that needs to be considered. This attack is possible regardless of what kind of authentication is used for the binding updates. In fact, the stronger and the more expensive the authentication protocol, the more serous this attack becomes.

Figure 6 shows how the attacker can induce authentic but unnecessary binding updates. When a spoofed packet sent by the attacker is tunneled to the mobile, the mobile typically responds by executing the binding update protocol with the claimed correspondent. The correspondent will eventually accept the binding update as both HoA and CoA are true. But the protocol execution is completely unnecessary. The attacker can repeat this with many different spoofed correspondent addresses to exhaust the resources of a single mobile, or with one spoofed correspondent address and many mobiles to attack a single correspondent.

Since the IP layer is stateless and BUs may be sent at any time, there is no practical way for the mobile or the correspondent to filter out the unnecessary binding updates without dropping also necessary ones. Therefore, the best defense against this attack is to limit the resources that the nodes allocate to processing binding updates to and from previously unknown mobiles. At worst, the attacker can prevent the use of route optimization.

8 Conclusion

We have described threats against the Mobile IPv6 route optimization protocol and protection mechanism used in the standard protocol. Some of the techniques are unconventional because the protocol needs to work globally without any global security infrastructure. Without such a solution, the Mobile IPv6 protocol would have been confined to intra-organizational use.

The experiences from the Mobile IPv6 design process highlight the need to consider early the potential security threats created by new technology. The some of the solutions described in this paper have been found to be applicable to other mobility protocols such as HIP [7]. It is promising that not only have the designers of newer protocols learned the specific protocol mechanisms but they have also started serious threat analysis and security design at an early stage in the design process.

References

1. Tuomas Aura and Pekka Nikander. Stateless connections. In *Proc. International Conference on Information and Communications Security (ICICS'97)*, volume 1334 of *LNCS*, pages 87–97, Beijing, China, November 1997. Springer.
2. Robert M. Hinden and Stephen E. Deering. IP version 6 addressing architecture. RFC 2373, IETF Network Working Group, July 1998.
3. Christian Huitema. *Routing in the Internet*. Prentice Hall, 1995.
4. David B. Johnson, Charles Perkins, and Jari Arkko. Mobility support in IPv6. Internet-Draft draft-ietf-mobileip-ipv6-24.txt, IETF Mobile IP Working Group, June 2003. Work in progress.
5. Gabriel Montenegro and Claude Castelluccia. SUCV identifiers and addresses. Internet Draft draft-montenegro-sucv-02.txt, IETF, November 2001. Work in progress.
6. Pekka Nikander. A scaleable architecture for ipv6 address ownership. Internet-Draft draft-nikander-ipng-pbk-addresses-00.txt, March 2001. Work in progress.
7. Pekka Nikander, Jukka Ylitalo, and Jorma Wall. Integrating security, mobility, and multi-homing in a HIP way. In *Proc. Network and Distributed Systems Security Symposium (NDSS'03)*, pages 87–99, San Diego, CA USA, February 2003.
8. Greg O'Shea and Michael Roe. Child-proof authentication for MIPv6 (CAM). *ACM Computer Communications Review*, 31(2), April 2001.
9. Vern Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *ACM Computer Communications Review (CCR)*, 31(3), July 2001.
10. Michael Roe, Tuomas Aura, Greg O'Shea, and Jari Arkko. Authentication of Mobile IPv6 binding updates and acknowledgments. Internet Draft draft-roe-mobileip-updateauth-01.txt, IETF Mobile IP Working Group, November 2001. Work in progress.
11. Christoph L. Schuba, Ivan V. Krsul, Markus G. Kuhn, Eugene H. Spaffold, Aurobindo Sundaram, and Diego Zamboni. Analysis of a denial of service attack on TCP. In *Proc. 1997 IEEE Symposium on Security and Privacy*, pages 208–223, Oakland, CA USA, May 1997. IEEE Computer Society Press.

Mobile IPv6 Security (Transcript of Discussion)

Tuomas Aura

Microsoft Research Ltd.

Ross Anderson: I can suggest another argument which is that the systemic attacks involve attackers who control the network near to the home. If I'm a mobile using this protocol I can try this correspondent, that correspondent, the next correspondent. They all send plain text keys to my home location. The only systemic attacker is somebody who can collect all of those. In other words, it's the GSM argument again; that you're providing something equivalent to the old-fashioned plain text wireline privacy.

Reply: So the design criteria for this protocol is that it should not be any worse than if we didn't have any mobility or any route optimisation.

Matt Blaze: It seems that there's a much simpler argument, which is that you'd have to be an active attacker in order to carry out this attack.

Reply: Yes, you'd have to be. In order to scoop binding updates, you always have to be an active attacker.

Matt Blaze: Since being an active attacker is already an attack on the routing infrastructure in some sense, essentially in order to carry out this attack you have to do what the attacker's trying to do in order to carry the attack out.

Reply: My most convincing argument is that the number of attackers is likely to be at most 30 on the route from one part to any other part of the Internet, and not any node on the Internet (as it was without authentication). We have dramatically reduced the number of potential attackers. Or, for any single router out there who would become an attacker, we have reduced the number of potential targets by a large amount.

Larry Paulson: I'd understand this better if I had a clear view of all the protocols in the stack. Now if I'm running a secure session with SSH or SSL, can I still be hijacked by these false binding updates?

Reply: Yes you can. End to end authentication prevents attacks against secrecy, but it won't prevent denial of service.

Most of the Internet traffic is unencrypted, or there's no end-to-end security. Any two nodes should be able to communicate, and this is to prevent a denial of service attack against them. If you are sending sensitive data then it's definitely a bigger problem if there are even a few nodes who can listen to your data.

Larry Paulson: If it is sensitive data, there are more powerful solutions which already work.

Reply: So what you would say is that if these two nodes - the correspondent and the mobile - have some kind of stronger relationship, or if they have a way of establishing a strong key to have an SSL connection, then they could get this key from there.

Ross Anderson: Or, if you look at it the other way around, somebody could use a strong key that they already have for application level traffic in order to authenticate the binding update.

Reply: We had versions of this protocol where you could do the same thing except that you wouldn't send this key because you already had a suitable key.

Ross Anderson: You could make a version that was compatible with everyday plain text, SSL encrypted or SSH encrypted traffic. In the binding update you send key 1, and in plain text to the home you send key 2, and then the key you use to authenticate is a hash of key 1 and key 2. In the case of the plain text key 2 is providing the authenticity, and in the case of SSL it is provided by key 1 in the binding update. A is *a fortiori* already encrypted and so that strengthens the security that we've got. One protocol toughens both up.

Reply: This is not always so easy. Usually the SSL connection is bound to the domain name and here the primary identifiers are IP addresses.

Ross Anderson: Oh no, that's a Not Invented Here argument. What I'm saying is that if you've got a secure tunnel, it doesn't matter why it's secure: SSL, SSH, somebody speaking in Albanian, whatever. For whatever reason, if the key K1 in the binding update happens to be unavailable to the attacker, but the key K2 from B to A is available, then having two keys which you hash together to provide your authentication sees you safe and sound.

Michael Roe: It does matter. If different sorts of keys are bound to identifiers in different spaces, then you might have a key that you know is good for the application because it's bound to that application, but you don't know which IP address that key is good for so you can't use it at the layer below.

Ross Anderson: Well it's unclear that using any old application key, for any old application running on that IP address, to reinforce the authentication already there could possibly introduce a vulnerability.

Reply: Can you postpone this for lunch because I'm just getting to the really interesting stuff that is new. [Laughter]

So, is this protocol good enough? It is simpler than some other proposed protocols that send the message directly to the current address and then to the home, so that the attacker has to intercept two messages. This works for a real

authentic mobile, but what they didn't take into account is that the attacker can decide the place of the care-of address. The binding update says "this is A, I'm at C;" so now we know that this really is A, but we don't know that A is at C. Does it cause problems if someone is lying about where they are? For example A, the mobile, is at home and subscribes to a data stream from some website, and then sends a binding update saying, "hello, I'm at C," and the data is sent there. Now we have a stream of unwanted data going to this address, and no matter how strong the end-to-end authentication is that you have between these two mobile correspondents, this attack is still possible unless you do something differently. Of course you could say that no acknowledgements are being sent but, for example, in the case of TCP, the beginning of the stream comes directly to A. A knows the sequence numbers, he can scoop the acknowledgements, and then you have to rely on ICMP error messages or similar, to stop the stream.

To make sure that mobile A is where he claims to be, we send the key as earlier, from home (it's now K0), and we send another key directly to the correspondent, to this current location. This is the test that the mobile can receive data that is sent to the current location; it's not absolute, but it seems to be as good a way as there is for testing that the mobile really is there, and the binding update is authenticated by using both of these keys.

In October, some of us got together and we noticed this problem. Quite soon after that no-one proposed any more protocols where they wouldn't take care of this somehow, usually by sending data to the current location and getting something back from there. Mike already talked about this, it's really a flow control or a consistency control problem. This is because B, before sending data here, should somehow know that this route can and wants to accept the data.

Ross Anderson: If you want to forward your mail, and have filled out a form at the Post Office, one of the things they do is send notifications to the old address as well to the new one.

Reply: The British post office does this. That's an interesting theory, we don't actually send anything to the old address.

Ross Anderson: You do, you send K0 to A.

Reply: OK. If you move from one C to another C, it's always you moving back home anyway. You give them your new temporary address, and the Post Office will notify both your permanent address and your new temporary address. And then the question is, is that good enough? We still need some improvements because of denial of service attacks. One is that the correspondent is stateful. After receiving the first message, it sends these two messages. It has to remember the K1 and K0, and we simply solve this by making the correspondent stateless by generating K0 and K1 in such a way that the correspondent doesn't need to remember them. The mobile has some kind of periodical changing secret which only he knows. He creates the keys K0 and K1 by hashing the home address

and current address together with his periodical changing secret, and this way he doesn't need to remember K0 and K1 for this mobile. It's an old trick for making the correspondent stateless.

We also worried about reflection, and amplification. I'm not sure how serious this is but it seems many people like Matt take it very seriously.

Matt Blaze: I don't take very much seriously.

Reply: Do you think it's a problem? The attack mode is like this: the attacker pretends to be C, sends the first message to the correspondent, and the correspondent sends back two messages through different routes to the current location. This is a way of amplifying a flooding attack by a factor of 2, and it is also a reflection in the way that this, at first address, is masked because these messages will not contain that address. And my solution to this was to just copy that procedure to these messages as an informational field, but actually the IP trace-back people are not happy about that because it is not easily accessed by routers.

So what do we do about this? Well, instead of sending one packet the mobile sends two packets; one from tunnel through the home, and another directly to the correspondent. He gets these two packets back, and now you know these message flows are balanced. You can even balance message length by adding rubbish to the first messages. Now there's no more reflection when you're running a protocol where the correspondent receives message 1A and he answers with 2A. Now, independently, he receives 1B and answers with 2B. So he receives a packet from an address and sends another packet back to that address. He doesn't need to correlate between these As and Bs, he can answer to this independently, and after that the mobile sends the authenticated binding update. This is the protocol in the mobile IP sixth draft.

Is it so important for reflection that we should really send this unnecessary packet through the hole? It might take longer than sending this packet directly, but that is now the way the protocol is specified. Some optimisations can, and probably will, be made. The mobile needs to send binding updates on two kinds of occasions; when he moves to a new address correspondent, and also to refresh the information to the cache. If he's refreshing a cache and there's lots of time for sending these packets, then it's minor, but if he really moves, then waiting for these packets to travel from home and back might actually be a quality of service problem. So maybe we allow reusing this K0 for a short while, but not forever.

Bruce Christianson: How atomic is a move? How much notice do you get that a move is imminent?

Reply: Sometimes the mobile knows. It really varies a lot. For example, if I'm connected to the office LAN and also have a wireless network and I disconnect

the office LAN, then the computer didn't get any notice that it should move over to the wireless network. On the other hand, when I go back to the same desk and plug in the LAN cable, the computer has all the time in the world. It can take its time, keep on using the wireless for as long as it wants, and then when it's ready, move the connections back to the office LAN.

Ross Anderson: You've created a business opportunity for somebody to provide high availability performance. Consider the disaster recovery plan with a typical investment bank. If your building burns down or whatever, your dealers get on to planes to your Paris office, with their laptops and their mobiles, and they do business there. But if your bank in London has burned down, or a home no longer exists, then you can't do that because with this protocol you can't roam. You'd have to start making, manufacturing and selling high quality infrastructure for homes that lack availability.

Reply: This is definitely a problem. This is something I don't like too much about mobile IP, that you have to have homes. For the average consumer that means you have to pay someone for maintaining your home. So there are alternative protocols. I'm not saying this is a smart way of doing mobility.

Matt Blaze: But the whole point of mobile IP is it gives you a stable address. If you don't need a stable address then...

Reply: Yes, and current client applications don't need a stable address.

Ross Anderson: Well perhaps the fix for this is that everybody will have three or four addresses.

Reply: Actually that's more advanced than it would be to move the home; freely adding new addresses and dropping old ones.

Pekka Nikander: Well, JI doesn't seem to be here but maybe I can just repeat what he says, and what I think also: that mobile IP is just a gross hack, and we are just planning to have a short but glorious life for it. There are better ways for doing mobility and it really involves separating the location and identifies aspects of the IP address.

Reply: The last attack, for which I don't have a very good solution, is an attack against any authentication protocol for binding updates. Our protocol is fairly resistant to this because it's a lightweight protocol: you send quite a few messages but there's no public key cryptography involved. But, if you use PKI and all that, then this attack is really bad. Here the attacker scoops a packet to the mobile at its home address S. It looked like it was coming from B, and the mobile says "Oh I received that, I don't see my packet from B, so let's do a binding update." It executes the binding update, no matter what the problem, and that will succeed because it's really the authentic C sending the binding update to B, and B will accept it, and there is nothing wrong with that. It was a true binding update, all the information was true, only the whole process was

completely unnecessary. The attacker can repeat this with multiple values of B to exhaust the mobile's resources, or he can repeat it with multiple mobiles of equal values of A to exhaust the correspondent's resources. The only way out of this seems to be just to limit the amount of resources you use for processing binding updates. If you feel that you're being exhausted by this work then give it up, don't do route optimisations, and tunnel everything through the home.

Richard Clayton: Since you've now got symmetry of the two paths it doesn't matter as much, but when you originally described the binding update you always showed that it was going directly from C across to the correspondent, rather than tunnelling them back and having them coming from A. One thinks of A as being more permanently connected, and perhaps having a lot more resources or computational power. There would be the possibility of doing much stronger protocols between A and the correspondent if the binding update had gone to A first.

Reply: There are some proposals for these kinds of protocols, where you move the heavy computation to the home agent. We actually had a public key version of this protocol, where you had the same things for denial of service with the key, but then the authentication would be done with CAM addresses and the address would be the hash of the public key. In that case a critical thing was that we could actually do all the public key computations at the home agent. Also, if the correspondent was mobile, you do the computations for the correspondent at the correspondent's home agent. I think it would never need to send any long certificates or public keys to the mobiles. But here, since it's all symmetric crypto, it doesn't make much sense to move anything to the home agent.

Concluding Discussion: Accounting for Resources

Chair: Bruce Christianson

Bruce Christianson: Perhaps I ought to introduce this discussion by saying what we do and don't mean by accounting for resources. The general observation is that if you make a resource available free then you're going to run out. So opinion has come and gone in the computing community, and particularly within the academic community, about whether or not it's moral, or a good idea, to charge (in some sense) for computing or network resources. And this usually falls on one of two horns of a dilemma, depending on whether you're talking about costing or pricing. Resources cost you the same amount whether anybody is using them or not, and generally they're there because you want people to use them rather than because you don't. And, if you're doing it to make money, then there's no incentive to fix any of the things that will lead to resources being wasted: if you discover there's a large amount of bandwidth being wasted because of fraud, you put up the prices and the honest customers pay up, and the problem is solved.

I should make clear that, when we talk about billing people, we don't mean actual folding money changing hands in the real world necessarily; what we mean is some kind of accounting for the resources being imposed, as a design discipline, when we put together the protocol that we're going to use. For example, you may have some kind of pre-payment, which may be in the form of your own CPU cycles. We've seen this in the past, in protocols[1] where you're given a puzzle to solve, and you have to send in a solution to the puzzle before you get past the first hurdle. It may be in the form of a deposit that you get back on successful completion of the protocol, and you can think of mobile computing as a little bit like this. So, the question is whether this idea, of putting information accounting for the use of resources into the design of the protocol, could be useful in helping the good guys enforce good behaviour? Particularly in the case of denial of service. I increasingly dislike that term because it's being stretched to cover all sorts of different things, but something which would allow us to starve out the grasshoppers[2] might be useful.

Virgil Gligor: I think we should have a proposition.

Bruce Christianson: I agree. The proposition is that when protocols are designed, they should include an accounting function that causes somebody, somewhere, to receive something a bit like a bill for the resources that the operations have used.

Virgil Gligor: I think this is an absolutely wonderful idea that will complicate matters significantly – so it better be worth it.

[1] "DOS-resistant Authentication with Client Puzzles", Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo, LNCS 2133, pp 170–178.

[2] or the locusts, if they're acting in concert.

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, pp. 235–241, 2004.

Matt Blaze: In particular, all of the mechanisms that I'm aware of to verify that the payment for accounting has been honestly transferred, are themselves more computationally and bandwidth intensive than most of the protocols that are exploited for denial of service. So I think doing this would require the invention of significant new kinds of cryptography, although it's possible that if those new kinds of cryptography were invented the problems would also go away.

Virgil Gligor: There is some clear evidence that accounting mechanisms are extremely complicated in real life; most phone companies dedicated years of research in billing.

Pekka Nikander: There is a big difference between billing and prepaid.

Ross Anderson: Sometimes you want to make the protocols more stateful to stop jamming or service denial, but in other cases – such as SYN Cookies – you want to make protocols stateless for the same reason, in order to prevent jamming, so it's not open and shut.

David Wheeler: Would statistical accounting, say one in a thousand, one in ten thousand, serve your purpose?

Bruce Christianson: On the whole, I think we're quite happy to play the odds...

Matt Blaze: I think quite the opposite, because if you only check one in a thousand times, then that means that 999 times out of a thousand the attacker is successful. Unless there's a penalty that is imposed on the dishonest user, statistical accounting doesn't work.

Ross Anderson: Well in that case it should be called statistical accountability. Think of IP trace-back.[3]

Bruce Christianson: The issue is whether somebody who does a bad thing once, and then never ever does it again, is actually a problem. If there are millions of them each doing one bad thing and I never succeed in catching any one of them, then potentially it is a problem.

David Wheeler: But if you check one in a thousand and there's a million of them, you'll catch a thousand.

Larry Paulson: I'm against the idea mainly because protocols are already so complicated, and it seems to be the wrong place to put in this sort of thing. Nothing is ever out there for a short time; if it's deployed it's going to be there a hundred years from now, with all the other stuff that's accumulated over the hundred years added to it. We shouldn't keep adding to the complexity of these things.

Richard Clayton: If we make it hard for the bad guys to sit at their machines and do bad things to the network because they run out of resources, all that happens is the bad guy will go around and find lots of innocent people, borrow their machines, and use their machines until *they* run out of resources. This results in denial of service (if you want to use that word) not only for what's been attacked, but also the people whose machines have been borrowed.

[3] "Practical Network Support for IP Traceback", Stefan Savage, David Wetherall, Anna Karlin and Tom Anderson, Proceedings of the ACM SIGCOMM Conference, August 2000.

Bruce Christianson: At the moment very often your machine *is* being used and you don't even realise that it is because you never get a "bill".

Richard Clayton: That's the idea behind these name and shame systems, to try and get all of the bad guys to exploit a small set of machines which are widely known to be compromised, because it concentrates the minds of people whose machine is involved.

Ernie Cohen: But you can limit that by simply not allowing any single machine to do a lot of damage. The whole point of the accountability is to just stop people doing drastically bad behaviour.

Ross Anderson: Sometimes drastically bad behaviour has socially good consequences, think of demonstrations. If issues are to be settled by peaceful actions rather than by civil war, you need some electronic equivalent of a million people turning up at Trafalgar Square and saying, we don't like this idea. It's interesting that in Britain the police have recently come to the conclusion that distributed denial of service attacks aren't illegal if the participants are participating voluntarily; if you can get a million people to send e-mails to 10 Downing Street that's legal.

Ernie Cohen: You get people to do denial of service on the website by actually going there and visiting it?

Ross Anderson: Yes, but it's not an attack, it's a demonstration: a lawful exercise of your civil rights.

Ernie Cohen: Yes, that's fine. But that's not the way that smurf attacks work. Smurf attacks work by exploiting somebody that has a relatively fat pipe and absolutely no control on sending gross nonsense.

Bruce Christianson: The notion that people who legitimately have resource might choose in concert voluntarily to spend it in a particular way is, I think as Ross says, a separate problem, and we're not trying to prevent that.

Alec Yasinac: A couple of years ago there was a paper on requiring you to maintain a secure system, so that you're not vulnerable to these zombies, that suggested a negative legal impact on you if you were caught as the one in one in a million.

Bruce Christianson: It also depends on what the consequence is when you're caught. In practice I guess the consequence might simply be that you're denied service for a period of time.

Alec Yasinac: Or maybe you're shot?

Victoria Stavridou: Do people typically do a complexity analysis on this sort of protocol? If so that would be a good basis to attribute the place from which the attack came without adding any additional complexity.

Ernie Cohen: Well, distributed protocols are normally assuming relatively honest participants. In a Byzantine protocol analysis you're not trying to count the possible number of messages that the bad guy could send.

Victoria Stavridou: But you know what the complexity is.

Ernie Cohen: You know what the complexity is, if people behave honestly, but we know that for regular crypto-protocols too. It's really trivial, except when people are doing nonsense things.

Victoria Stavridou: I guess then Bruce's point was, is there some way we could account for legal activities as opposed to illegal activities?

Ernie Cohen: You could.

Bruce Christianson: Many protocols begin with an unsolicited first message. Clearly there's a problem there. Everybody can send an unsolicited first message.

Pekka Nikander: If we look at the history of warfare, in the early days armies were able to move as long as they could steal their resources from the surrounding farmers. I think that's more or less the situation in the network now. You are able to launch these attacks because you can rob the resources from where they are, and if we could limit the amount you can rob maybe we can change the rules of the game.

Bruce Christianson: Some of the old network congestion control protocols were like this in essence. Particular areas of the network had a certain capacity, and if they weren't full then empty packets would circulate so that the actual traffic load was constant. And the rule was, if you wanted to put something into that area of the network, then you had to start by taking an empty packet out and substituting your packet for it (and there were rules about how it had to be a packet of the correct colour, and how you recirculate empties and so forth). Now the question is whether you could enforce a protocol like that in a way that puts the computational burden on the bad guy.

Virgil Gligor: Continuing Pekka's analogy, what happened in the middle ages was that defenders, villagers, would just retreat and burn everything; they were destroying their own infrastructure for a while so as to starve the attackers.

Alec Yasinac: The model I was pointing out a few minutes ago is that you make it an incentive for the people who have the resources not to allow that resource to be stolen but to destroy it first. One of the reasons that technique was abandoned by the military, is because war is a political tool. If you destroy or consume all the resources, you'll never win the people back. The same could be true here.

Wenbo Mao: If you require some evidence of work, then you must have a way to check it otherwise it would be disastrous. You need some magic weapon, such that the client accounting work is orders of magnitude more difficult than the way you check. Otherwise they'll just send rubbish. If you don't check, then sending rubbish is easy.

Bruce Christianson: Yes, at the moment it's really easy to just send rubbish and nobody's sponsoring you, nobody's vouching for you.

Wenbo Mao: But, if you start a check then there is automatically a denial of service attack, unless you have an extremely asymmetric way of verifying this evidence.

Richard Clayton: If you have asymmetry, it's often very hard to see which way round it should be. We started off with a view that the clients were very computationally challenged when talking to web servers which were going to do all the computational work in order to do secure tunnels. In fact it's turned out

the other way round. Everybody's got a 2 GHz Pentium, and the poor old web server is struggling in order to be able to give a service.

Ernie Cohen: The thing that I've always found completely shocking is that in the place where everybody sees denial of service every day, namely spam, the most obvious mechanism, which is to charge people when they send a message, hasn't taken off even though people have talked about it for over five years.

Pekka Nikander: Well there are some forms of social charging. There are systems where when you get the message from somebody that sends an automatic reply saying please send it again. So if you get a message from somebody that you don't know, it never arrives to your mailbox, but the robot will answer that person saying, if the message is sincere please send it again. That's a way of social charging.

Ernie Cohen: But that's also denial of service. And you've wiped out a legitimate form of broadcast.

Bruce Christianson: There's an arms race here. What will happen is you'll get a cookie in the reply, because the robot will send a cookie and say, please re-send the message with this cookie.

Ernie Cohen: But the problem is that's only effective if you can stop a robot from sending the repeat message. Anyway, there's nothing wrong with mass mailing if somebody's willing to pay some money to get a bit of shelf space.

Tuomas Aura: There's a kind of protocol against telemarketers, I'm not sure if it will apply to IP networks so easily, but in the telephone network there's a charging infrastructure. What could be done is that when you want to call me, you have to pay $10 upfront. At the end of the call I can press a key on the phone to keep this $10. If I don't press then you never pay, but I can keep it if I want. And of course, if it's someone I might want to call me again I probably wouldn't, because I don't want to upset them, but if it's a telemarketer I make $10, nice talking to you.

Ross Anderson: If I was a phone company I would consider offering people the possibility of having their directory phone number as a 0900 number so that anybody who looks up their number in the phone directory and called them would be paying them 40p a minute, and also 40p a minute to the phone company. To people you really like, you give your ex-directory low cost to call number. This firstly sorts out the telemarketing problem (except people who are really, really serious), secondly it multiplies the revenue to the phone companies (so there's some chance it might actually get done), and thirdly it solves the problem you have in places like Los Angeles where almost no numbers nowadays are in the directory because people are so fed up with marketing calls.

Matt Blaze: One of the problems with a lot of these "demonstrate that you've done something expensive" protocols, for example hash collisions and so on, that are easy to verify but expensive to do, is that while you are causing some payment in computation to be made, it's not *useful* computation for anyone. What you're essentially doing is the equivalent of saying, I will talk to you on the phone if you demonstrate to me that you have burned $10. What we end

up with is this spiral in which more and more resources get burned. A challenge would be to find some efficient way of transferring something of useful value.

Bruce Christianson: We're in danger of being driven into a sort of scorched earth policy, where we launch a massive distributed denial of service attack against ourselves in order to prove that we are sincere.

Virgil Gligor: The biggest problem with solving puzzles is that it does not guarantee access to the people who burn the cycles, who may in fact be paying twice and still not get access to the server. I find that a particularly bad idea. I could agree with burning cycles if that would guarantee access, but it only guarantees that you have access if everyone is playing the protocol by the rules, and not everybody would.

Ernie Cohen: It seems like the main thing as an enabling technology for all of this, would be to have a public electronic cash infrastructure, as opposed to a public key infrastructure.

Pekka Nikander: I think Ross's idea can be applied to the Internet. In IPv6 you can have a random address. So you can put an address in the directory which is basically your real address in an encrypted form, or some form that means you have to burn cycles to learn the real address. Once you learn the real address then you can connect directly.

Geraint Price: There's a slight problem with communication networks and computers because you've got layering of resources. If you charge at one level then I hit you at the level below. The lower the levels get, the less you want to be able to charge, because of the amount you have to ship onto the levels. You just end up hitting the problem.

Bruce Christianson: One of the arguments for having mediated access, in other words saying that a random client cannot simply go to a random server and send it a message, is that this effectively pushes you up through the protocol stack at the mediative point, and then you can have a handoff where somebody's acting as a broker for two sets of charges.

Ross Anderson: We're not considering the stack to be tall enough, and in the case of something like spam, we're looking at an artificially restricted part of the problem. The overall problem, as Herbert Simon pointed out[4], is that as we move into a world in which everything is plentiful except people's attention, we're going to get an attention economy. Now what this is going to mean is that people are forever trying to grab attention, and they're trying to externalise the cost (which is anything that involves them paying attention themselves). For example, I increasingly get requests to review papers, give talks, do other academic things, which are sent to me not by the professor who wants me to do this thing, but by his secretary. And for the last few years I have an almost invariable rule that when I get a request from a secretary, I tell them politely to get stuffed, because if a professor cannot be bothered to spend 30 seconds with

[4] "Designing Organizations for an Information-Rich World", Herbert A. Simon *Computers, Communications and the Public Interest* (ed. Martin Greenberger), The Johns Hopkins University Press, Baltimore, 1971, pp 40–41.

his own fingers on his own keyboard to send an email why should I spend more than the one second it takes to say no.

While we can build partial solutions for anti-jamming at various levels in a protocol stack, we may be working against the grain of human behaviour by trying to solve the symptoms rather than the problem. From the point of view of the human who is participating in the attention economy, what we really need are the tools that empower us to leverage our own precious minutes of conscious existence, rather than be cramped or bundled or bought and sold by the people who purchase the implementation of fancy new systems, and who themselves are trying to exercise power over us. This is perhaps what's missing.

Back to the Beginning

Roger Needham: Of course, in Computer Science, good research is done with a shovel, not with tweezers ...

B. Christianson et al. (Eds.): Security Protocols 2002, LNCS 2845, p. 242, 2004.

Author Index